Caché ObjectScript $ZUTIL Reference

Version 2010.1
14 October 2009

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

Caché ObjectScript $ZUTIL Reference
Caché Version 2010.1 14 October 2009

This book was assembled and formatted in Adobe Page Description Format (PDF) using tools and information from the following sources: Sun Microsystems, RenderX, Inc., Adobe Systems, and the World Wide Web Consortium at www.w3c.org. The primary document development tools were special-purpose XML-processing applications built by InterSystems using Caché and Java.

InterSystems CACHÉ® and InterSystems ENSEMBLE®

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Customer Support

Tel:	+1 617 621-0700
Fax:	+1 617 374-9391
Email:	support@InterSystems.com

Table of Contents

About This Book

This book provides reference material for the Caché ObjectScript $ZUTIL functions. As of version 2010.1, those functions not obsoleted in version 2010.1 by changes to the system were deprecated and removed from the documentation. This book serves only to isolate the documentation of the remaining functions.

This book contains the following section:

- $ZUTIL Functions

Other related topics in the Caché documentation set are:

- *Caché ObjectScript Reference*
- *Using Caché ObjectScript*
- *Using Caché Objects*, particularly the chapters "Using Objects with Caché ObjectScript" and "Object-specific ObjectScript Features".
- *Caché I/O Device Guide*

For general information, see *Using InterSystems Documentation*.

$ZUTIL Functions

The following are reference pages for the **$ZUTIL** utility functions. Many of these functions are used to return or set configuration, locale, and platform parameters, and to support Unicode characters. These functions supplement the Caché ObjectScript general functions described in the Caché ObjsctScript Reference document. Functions are listed in alphabetical order.

$ZUTIL(4)

Terminates a Caché process.

```
$ZUTIL(4,pid,flag)
$ZU(4,pid,flag)
```

Parameters

pid	The Process ID (pid) of the Caché process you wish to terminate.
flag	*Optional* — A positive or negative integer. A negative integer flag governs process termination operations. You can specify a value of -65 to force termination when the process to be terminated is a system daemon, or is in an unresolved transaction state. Other negative integer values are reserved for internal use only. A positive integer is used as an exit status code that Caché returns to the host operating system upon termination. These values can be used in shell procedures. The standard values for exit status codes are 0 for UNIX® and Windows systems, and 1 for OpenVMS systems.

Description

The **$ZUTIL(4)** function causes the specified process to terminate and returns execution to the operating system command level (OpenVMS DCL command level, UNIX® shell or Windows program manager). **$ZUTIL(4)** returns a numeric value indicating success or failure (see table below). When a process issues **$ZUTIL(4)** against itself, **$ZUTIL(4)** completes execution, including issuing its return code, before terminating the process.

One common reason for failure of **$ZUTIL(4,pid)** is that the specified *pid* is the Process ID of a Caché system daemon process. To terminate a system daemon process, you can specify **$ZUTIL(4,pid,-65)**. This optional -65 option permits termination of system daemon processes.

The optional -65 option also has an effect on termination processing of non-daemon processes. If a process is a halt or rollback state, issuing **$ZUTIL(4,pid)** returns the -4 error code, and transaction integrity is maintained. Issuing **$ZUTIL(4,pid,-65)** terminates the process; Caché does not roll back the open transaction, but does release all transaction locks. This may leave a transaction in an inconsistent state.

Return Values

$ZUTIL(4) returns the following values:

Value	Meaning
1	Success. **$ZUTIL(4)** successfully caused the process to exit.
0	Failure. The process is a system daemon process, such as Write Daemon. Neither **$ZUTIL(4)** nor the RESJOB utility (which uses it) can terminate a system daemon, unless you specify **$ZUTIL(4,pid,-65)**.
-1	Failure. The process is non-responsive.
-2	Failure. The process is inactive, a ghost or "dead" process.
-3	Failure. The process is not a Caché process. It is not on the Caché job table.
-4	Failure. The process has an open transaction that is currently in a halt or rollback state and cannot be terminated unless you specify **$ZUTIL(4,pid,-65)**.

Examples

The following example causes process number 1584 to exit:

```
SET procid=1584
SET x=$ZUTIL(4,procid)
IF x=1 { WRITE "Success: ",procid," terminated" }
ELSEIF x=0 {
   WRITE "Failure: ",procid," is a daemon",!
   WRITE "Use the $ZUTIL(4,pid,-65) syntax" }
ELSEIF x=-3 { WRITE "Process ",procid," not recognized" }
ELSEIF x=-4 { WRITE "Process ",procid," in halt or rollback" }
ELSE { WRITE "Failure: $ZUTIL(4) returned ",x," for ",procid }
```

Notes

To determine the process IDs of all currently running processes, go to the System Management Portal, and select **[Home] > [Processes]**.

OpenVMS: **$ZUTIL(4,pid)** is the internal function called by the RESJOB utility.

See Also

- HALT command
- JOB command

$ZUTIL(5)

Returns current namespace or switches to another namespace.

```
$ZUTIL(5,nspace)
$ZU(5,nspace)
```

Parameters

nspace	*Optional* — Namespace to be made current, enclosed in quotation marks. Can be a namespace name or an implied namespace.

Description

This function can be used to return the process' current namespace or to switch the current namespace to another namespace:

- To return the process' current namespace, use **$ZUTIL(5)** with no argument.
- To change the current namespace, use **$ZUTIL(5,nspace)** with the namespace name as the second argument.

After you issue **$ZUTIL(5,nspace)**, Caché draws routines and globals from the Caché database in this new namespace. Your effective working namespace also changes.

Parameters

nspace

This parameter specifies the namespace to be made current. Enclose the namespace name in double quotes. The namespace you specify can be an explicit namespace or an implied namespace. An implied namespace is a directory path or OpenVMS file specification preceded by two caret characters: either "^^*dir*" (on the current system), or "^*system*^*dir*".

Namespace names are not case-sensitive. Caché always displays explicit namespace names in all uppercase letters, and implied namespace names in all lowercase letters.

Notes

$ZUTIL(5,nspace) retains the information Caché needs to return from routines in the new namespace. When a routine in the new namespace quits, it returns correctly to the routine that called it, which may return to its caller in the same or other namespaces, and so on. Subsequent routine calls, however, continue to refer to the namespace named in **$ZUTIL(5,nspace)**.

Thus, if you issue **$ZUTIL(5,nspace)** and then issue **OPEN** for a sequential file without specifying a namespace, the operating system either:

- Finds the file in *nspace*
- Creates the file in *nspace* if the file does not exist

If you issue **$ZUTIL(5,nspace)** in a subroutine, the original routine finishes; however, routines called are taken from the new namespace, *nspace*. When you halt from Caché, your namespace reverts to the one you were in when you entered Caché.

If you incorporate **$ZUTIL(5,nspace)** into a library routine from the system manager's namespace (**%SYS**), you give all programmers access to all namespaces regardless of their UIC or privilege level.

When using **$ZUTIL(5)** to change the current namespace, Caché clears the global vectors. However, when issuing a **$ZUTIL(5,nspace)**, where *nspace* is the current namespace (in other words, not changing the namespace), Caché does not clear global vectors. If you wish to clear global vectors without changing the current namespace, use **$ZUTIL(69,43,1)** before calling **$ZUTIL(5)**.

If you wish to have a process reference the routines in another namespace, without changing the current namespace, use **$ZUTIL(20)** or **$ZUTIL(39)**.

$ZUTIL(5) and Implied Namespaces

Both **$ZUTIL(5)** and the **ZNSPACE** command can be used to change the current namespace to an implied namespace, with USER as the default database. However, their mapping of this implied namespace differs. This difference affects the user's ability to locate and run % routines such as %SS, %GD, and %RI, and to access some % globals, such as those whose names begin with %q.

- **$ZUTIL(5)** does not create additional default mappings from the implied namespace. A process cannot find and execute % routines and % globals that are physically located in the CACHELIB database. Attempting to access these % routines results in a <NOROUTINE> error. Attempting to access these %q globals results in an <UNDEFINED> error.
- **ZNSPACE** creates additional default mappings from the implied namespace. These mappings are the same as for a normal (explicit) namespace. They allow a process to find and execute the % routines and % globals that are physically located in the CACHELIB database.

Setting the **$ZNSPACE** special variable is the same as issuing a **ZNSPACE** command.

Changing Namespaces within Application Code

Object and SQL code assumes that it is running in a single namespace; hence, changing namespaces with open object instances or SQL cursors can lead to code running incorrectly. Typically, there is no need to explicitly change namespaces, as the various Object, SQL, and CSP servers automatically ensure that application code is run in the correct namespace.

Also, changing namespaces demands a relatively high amount of computing power compared to other commands; if possible, application code should avoid it.

See Also

- ZNSPACE command
- $ZUTIL(20) Specify namespace to search for routines function
- $ZUTIL(39) Specify namespace to search for % routines function
- $ZUTIL(68,43) Disable or Enable Clearing of Global Vectors for the Current Process function
- $ZUTIL(69,43) Disable or Enable Clearing of Global Vectors System-wide function
- $ZUTIL(90,10) Test Whether a Namespace is Defined function
- $ZNSPACE special variable
- Configuring Namespaces in *Caché System Administration Guide*

$ZUTIL(9)

Broadcasts a message to a specified device.

```
$ZUTIL(9,terminal,message,timeout)
$ZU(9,terminal,message,timeout)

$ZUTIL(9,"",message,loglevel)
$ZU(9,"",message,loglevel)
```

Parameters

terminal	A terminal name, specified as a quoted string, or the null string.
message	The message to send, specified as a quoted string.
timeout	*Optional* — An integer specifying the number of seconds to wait before timeout.
loglevel	*Optional* — A numeric code indicating which logging level option to use.

Description

$ZUTIL(9) has two syntactic forms. The first sends a message to the specified terminal, and can optionally time out. The second sends a message to the operator console (specified by the null string), and also logs the message.

- **$ZUTIL(9,"",*message*)** passes the message to the operator console, logging it in the console log file. By default, the console log file is cconsole.log, which can be accessed via the System Management Portal System Logs option. This default console log file location is configurable. Go to the System Management Portal, select **[Home] > [Configuration] > [Memory Settings]**. View and edit the current setting of **ConsoleFile**. By default this setting is blank, routing console

messages to cconsole.log in the MGR directory. If you change this setting, you must restart Caché for this change to take effect.

- **$ZUTIL(9,*terminal*,*message*)** passes the message to the specified terminal. If you specify your own principal device, your message appears on your terminal screen.

$ZUTIL(9) does not add any carriage control to the message it sends. To include any carriage control (carriage returns or line feeds), you must include them in the message, using **$CHAR(10)** and **$CHAR(13)**.

Parameters

terminal

The device name of the terminal to which you want to send a message, specified as a quoted string. Specify the null string ("") to send the message to the system console.

message

The message to send, specified as a quoted string.

timeout

Optional (used with named terminal only) — A timeout in seconds. If **$ZUTIL(9)** is not able to send the message during the period of the timeout, it ceases attempts to send the message after the timeout expires.

loglevel

Optional (used with operator console only) — The log level you want to assign to the message. You can use the following values:

- 0 = Send the message to the following locations: Operator console log file, Caché console.
- 1 = Send the message to the following locations: Operator console log file, Caché console, System-wide operator console facility. 1 is the default.

Examples

In the following example, **$ZUTIL(9)** sends the message GOOD MORNING on a separate line to the terminal /dev/tty07. (**$CHAR(13)** is a carriage return, **$CHAR(10)** is a line feed.)

```
SET X=$ZUTIL(9,"/dev/tty07",
   $CHAR(13)_$CHAR(10)_"GOOD MORNING"_$CHAR(13)_$CHAR(10))
```

In the following example, **$ZUTIL(9)** sends the message you specify on a separate line to the operator console. (**$CHAR(13)** is a carriage return, **$CHAR(10)** is a line feed.)

```
READ "Operator message? ",mymess,!
SET X=$ZUTIL(9,"",$CHAR(13)_$CHAR(10)_mymess)
```

Notes

$ZUTIL(94) complements **$ZUTIL(9)**. **$ZUTIL(9)** sends a message to a *specified device* while **$ZUTIL(94)** sends a message to the *principal device of a specified process*. Be sure that you use the right function for the right purpose. If you send a process id to **$ZUTIL(9)** or a terminal device name to **$ZUTIL(94)**, you receive a <FUNCTION> error.

To view messages in the Console Log File, go to the System Management Portal. Select **[Home] > [System Logs] > [View Console Log]** to display the log contents. The example above (with carriage return / line feed) writes a message such as the following in the Console Log:

```
02/24-10:40:13:547 ( 2444) 0
GOOD MORNING
```

A long message to the Console Log File may be truncated. The truncation length of a Console Log message depends upon both the version of Caché and the operating system platform, as shown in the following table:

	Windows	UNIX® and VMS
Caché 2009.1 and subsequent versions	2700 characters	1024 characters
Caché versions prior to 2009.1	2700 characters	400 characters, minus the datetime stamp, pid, and severity level

See Also

- $ZUTIL(94) Broadcast a Message to a Specified Process function
- Terminal I/O in *Using Caché ObjectScript*

$ZUTIL(12)

Converts file or directory name to canonical form.

```
$ZUTIL(12,name,canon,case)
$ZU(12,name,canon,case)
```

Parameters

name	*Optional* — A string expression specifying a directory name, namespace name, or OpenVMS logical device name to be canonicalized. If not specified, the default is the system manager's directory.
canon	*Optional* — An integer flag that specifies how conversion to canonical form is performed. Available value are 0, 1, 2, and 3. The default is 0 on OpenVMS, and 1 on other operating systems.
case	*Optional* — Windows only: A boolean flag that specifies whether to return *name* in all lowercase letters or to preserve uppercase letters. 1 = preserve uppercase letters in existing portion of pathname as stored, and in non-existing portion of pathname a specified in *name*. 0 = convert *name* to all lowercase letters (the default).

Description

$ZUTIL(12) converts the filename or OpenVMS logical device name specified by the expression *name* to canonical form. It expands a partial pathname to a full pathname for a specified file or directory, the current directory, or the current namespace.

You can use the wildcard options of the **$ZSEARCH** function to locate an existing directory and return its full pathname. You can use **$ZUTIL(140)** to create a new directory, or to return information about an existing file or directory.

Directory Pathnames

The following rules govern the expansion of directory pathnames. (These examples assume that Caché is installed on disk "c:" and that "fred" is a nonexistent directory):

- **$ZUTIL(12)** (with no arguments) returns the current system manager's directory pathname. For example: c:\InterSystems\Cache\mgr\.

- **$ZUTIL(12,"")** or **$ZUTIL(12,"c:")** returns the current namespace pathname. For example: c:\InterSystems\Cache\mgr\user\.
- **$ZUTIL(12,"c:\")** returns the disk path: c:\. Note that a disk path is returned even if the specified letter does not correspond to a defined disk.
- **$ZUTIL(12,"fred")** returns the current namespace pathname for the specified directory. For example: c:\InterSystems\Cache\mgr\user\fred\.
- **$ZUTIL(12,"\fred")** returns the specified directory as a top-level directory. For example: c:\fred\.
- On Windows systems, pathnames can use a single dot (.) to indicate the current directory, or a double dot (..) to indicate its parent directory. Thus **$ZUTIL(12,".")** returns the current namespace directory: c:\InterSystems\Cache\mgr\user\, and **$ZUTIL(12,"..")** returns the parent directory of the current namespace directory: c:\InterSystems\Cache\mgr\.
- On OpenVMS systems, pathnames can use a single dot (.) to indicate the current directory, or a hyphen within brackets ([-]) to indicate its parent directory. Thus **$ZUTIL(12,".")** returns the current namespace directory: _VMSIT3$DKA0:[MAURICE.365.MGR.USER], and **$ZUTIL(12,"[-]")** returns the parent directory of the current namespace directory: _VMSIT3$DKA0:[MAURICE.365.MGR].

Note: For OpenVMS *name* cannot include a VMS node name. Therefore, **$ZUTIL(12)** can only detect existing VMS directories that exist on the local node. If *name* includes a VMS node name, **$ZUTIL(12)** considers it an invalid directory name and returns null.

The canon Flag

$ZUTIL(12,*name*,*canon*) determines the action to be performed based on the value of *canon*. Valid values are:

0	Canonicalize *name* as a filename. Caché does not check that *name* is a valid filename, but may check capitalization and compression of higher-level directory names (see below). Thus: c:\InterSystems\Cache\mgr\user\fred.
1	*Windows*: Canonicalize *name* as a directory name. Caché checks that *name* is a possible directory name—no existing file already has that name. If *name* is the pathname of an existing file, **$ZUTIL(12,*name*,1)** returns the empty string. Otherwise, **$ZUTIL(12,*name*,1)** returns *name* as a directory name. Caché does not check that *name* is an existing directory name, but may check capitalization and compression of higher-level directory names (see below). Thus: c:\InterSystems\Cache\mgr\user\fred\ whether or not the directory “fred” exists. This is the default. *OpenVMS*: Canonicalize *name* as a directory name. Returns null if it is an invalid directory name (existing or not). *UNIX*®: Canonicalize *name* as a directory name. Returns null if it is not a directory (existing or not). For example, if *name* is a file or a special device such as a raw partition.
2	If *name* refers to an existing directory, canonicalize *name* and return the full pathname. Otherwise, return null.
3	If *name* refers to an existing directory or a special device that can contain a Caché database, canonicalize *name* and return the full pathname. Otherwise, return null.

The case Flag

On Windows systems, the default canonical form returned is entirely in lowercase letters, regardless of the case of *name*. Because Windows and Caché ObjectScript are not case-sensitive, this is usually irrelevant. However, this may affect some Windows applications (such as Java) which are case-sensitive.

To preserve uppercase letters on Windows systems, specify a *case* flag with a value of 1. This has the following effects:

- The drive letter is always converted to uppercase.
- If the directory or file you specified does not exist on your system, **$ZUTIL(12)** follows the capitalization you specified. Letters specified as uppercase remain in uppercase.
- If the directory or file exists on your system, **$ZUTIL(12)** follows the capitalization conventions of the directory itself, rather than the capitalization in the string you specified.

For example, if you used the default installation location for Caché:

```
SET x=$ZUTIL(12,"\iNtERSyStEms\cAcHe\mGr\vIsiToR\fReD",0,1)
WRITE x
```

The above program returns: C:\InterSystems\Cache\Mgr\vIsiToR\fReD\, deriving the case of the existing directories in this pathname (C:\InterSystems\Cache\Mgr\) from the file system, and deriving the case of the nonexistent directories (vIsiToR\fReD\) from the input string.

The *case* flag is accepted on non-Windows systems, but performs no operation.

If the pathname contain one or more Windows 8.3 compressed directory names, **$ZUTIL(12)** will expand these names if the *case* flag is set to 1. This is shown in the following example:

```
WRITE !,$ZUTIL(12,"c:\INTERS~1\Fred",0,0)
  ; compressed filename not expanded
WRITE !,$ZUTIL(12,"c:\INTERS~1\Fred",0,1)
  ; compressed filename expanded to
  ; C:\InterSystems\Fred
```

Examples

The following example returns the canonical form pathname of the system manager's current directory and the current namespace. Note that on Windows systems, canonical forms are all lowercase:

```
WRITE !,$ZUTIL(12)     ; current mgr directory
WRITE !,$ZUTIL(12,"")  ; current namespace
```

The following example returns the canonical form of the pathname of the nonexistent file "Fred", and the invalid filename "Fr@d":

```
WRITE !,"flag 0: ",$ZUTIL(12,"Fred",0),!,$ZUTIL(12,"Fr@d",0)
WRITE !,"flag 1: ",$ZUTIL(12,"Fred",1),!,$ZUTIL(12,"Fr@d",1)
WRITE !,"flag 2: ",$ZUTIL(12,"Fred",2),!,$ZUTIL(12,"Fr@d",2)
WRITE !,"flag 3: ",$ZUTIL(12,"Fred",3),!,$ZUTIL(12,"Fr@d",3)
```

Flag 0 returns a canonical form for "Fred" and "Fr@d" as filenames. Flag 1 returns the canonical form for "Fred" and "Fr@d" as directory names. Note that filename validation is not performed. Flag 2 and Flag 3 return the null string because "Fred" does not exist.

In the following UNIX® example, if /us/bill/ is a valid name, then:

```
WRITE $ZUTIL(12,"^^/us/bill//",2)
```

returns and canonicalizes the reference, treating it the same as /us/bill/.

See Also

- $ZSEARCH function
- $ZUTIL(140) Return File Attributes, Create or Delete a Directory function
- Configuring Caché in *Caché System Administration Guide*

$ZUTIL(15)

Converts RMS filename to canonical form.

```
$ZUTIL(15,devname)
$ZU(15,devname)
```

Parameters

devname	A RMS filename.

Description

OpenVMS: Use this form of **$ZUTIL** to convert the RMS file name specified by *devname* to canonical form.

Example

```
WRITE $ZUTIL(15,LOGIN.COM)
```

returns LARRY$DUA0:[SYSM.NEW]LOGIN.COM;1

See Also

- $ZUTIL(12) convert filename to canonical form function
- RMS and Sequential File I/O in the *Caché I/O Device Guide*

$ZUTIL(18)

Sets undefined variable handling for the current process.

```
$ZUTIL(18,n)
$ZU(18,n)
```

Parameters

n	*Optional* — A numeric code that determines how Caché treats an undefined variable. Permitted values are 0, 1, and 2. If you omit *n*, **$ZUTIL(18)** returns the current value without changing it.

Description

Use **$ZUTIL(18)** to define how undefined variables are to be handled in the current process. The value you specify for *n* determines the behavior of Caché when it encounters an undefined variable. Setting *n* changes this behavior for local, process-private global, and global variables; it has no effect on special variables.

By default, if a process makes reference to an undefined variable, it produces an <UNDEFINED> error. This system-wide default behavior is configurable. Go to the System Management Portal, select **[Home] > [Configuration] > [Compatibility Settings]**. View and edit the current setting of **Undefined**. The default is 0 (always throw an error). You can override this default system-wide by using the $ZUTIL(69,0) function.

$ZUTIL(18) only affects variables invoked by the current process. **$ZUTIL(69,0)** only affects variables invoked by subsequent processes; it does not affect the current process.

You can call **$ZUTIL(18)** to return the current setting. When calling **$ZUTIL(18,n)** the value returned is the previous setting, not the value set by the function call.

You can use the **$GET** function to return a default value when a specified variable is undefined. Setting **$ZUTIL(18)** has no effect on **$GET**. You can use the **ZWRITE** command to display defined variables that have a null string value; **ZWRITE** does not display undefined variables that return a null string.

MVBasic Undefined Variables

Caché MultiValue Basic (MVBasic) routines are conditionally independent in their handling of undefined variables:

- By default, MVBasic undefined variables are controlled by the **$ZUTIL(18)** and **$ZUTIL(69,0)** settings. They are handled exactly the same as regular Caché variables.
- If **$ZUTIL(68,72,0)** is set for the current process, or **$ZUTIL(69,72,0)** is set system wide, MVBasic undefined variables in the current process are controlled by the **$ZUTIL(18)** setting.
- If **$ZUTIL(68,72,1)** is set for the current process, or **$ZUTIL(69,72,1)** is set system wide, MVBasic undefined variable behavior is independent of the **$ZUTIL(18)** and **$ZUTIL(69,0)** settings.

For further details, refer to the Variables page of the *Caché MVBasic Reference*.

Parameters

n

A numeric code that specifies how Caché treats an undefined variable in the current process:

0	Issues the <UNDEFINED> error for any undefined variable.
1	Returns a null string for a reference to an undefined variable with subscripts, and issues the <UNDEFINED> error for undefined variables without subscripts.
2	Returns a null string (instead of issuing an <UNDEFINED> error) for any undefined variable.

Integers larger than 2 or smaller than –1 are ignored as no-ops. A value of –1 sets **$ZUTIL(18)** to 2.

Notes

The **$DATA** function tests if a specified variable is defined. It returns 0 if the variable is undefined.

The **$GET** function returns a default value if a specified variable is undefined. The basic form of **$GET** returns a null string if the specified variable is undefined.

The **$ZUTIL(18)** function defines handling behavior for *all* undefined variables: local variables, process-private globals, and globals. Setting **$ZUTIL(18)** has no effect on **$DATA** or **$GET**.

You can use the **ZWRITE** command to display *defined* variables that have a null string value; **ZWRITE** does not display undefined variables that return a null string.

Examples

In the following example, Caché by default issues an <UNDEFINED> error when encountering an undefined variable. **$ZUTIL(18)** overrides that default for the local process.

```
WRITE !,"system:",$ZUTIL(69,0)," local:",$ZUTIL(18)
SET x=$ZUTIL(18,2)
WRITE !,"system:",$ZUTIL(69,0)," local:",$ZUTIL(18)
NEW fred
IF fred="" {
   WRITE !,"null string for undefined variable" }
ELSE {
   WRITE !,$GET(fred,"undefined variable not null") }
```

In the following example, **$ZUTIL(18,1)** sets undefined variable handling to return a null string for defined subscripts (such as ^||X(A,B,C)) of an undefined process-private global ^||X. The **ZWRITE** command is used to show variables set to a null string.

```
SET A="Fred",B="Barney",C="Wilma"
WRITE !,$ZUTIL(18,1)
WRITE !,"system:",$ZUTIL(69,0)," local:",$ZUTIL(18),!
SET D=^||X(A,B,C)
SET E=^||X(1,2,"HELLO")
ZWRITE D,E
WRITE !,$GET(^||X,"^||X is not defined")
```

In the following example, **$ZUTIL(18,2)** returns a null string for the undefined variable J.

```
WRITE !,$ZUTIL(18,2)
WRITE !,"system:",$ZUTIL(69,0)," local:",$ZUTIL(18)
WRITE !,$GET(J,"Variable is undefined")
WRITE !,J,"No UNDEFINED error issued"
```

See Also

- $ZUTIL(69,0) Set Undefined Variable Handling System-wide function
- $ZUTIL(68,72) Set MVBasic Undefined Variable Handling function
- $ZUTIL(69,72) Set MVBasic Undefined Variable Handling System-wide function
- ZWRITE command
- $DATA function
- $GET function

$ZUTIL(20)

Specifies the namespace(s) that contains the routine dataset.

```
$ZUTIL(20,new,second,third)
$ZU(20,new,second,third)
```

Parameters

new	Specify the primary namespace that contains the routine dataset.
second	*Optional* — Specify the second namespace to search for routines.
third	*Optional* — Specify the third namespace to search for routines.

Description

To change the source from which you want to draw routines, use this form of **$ZUTIL** to specify the namespace that contains the routine dataset in which the routines you want to use are located.

Enclose the namespace in double quotes. The namespace you specify can be either an explicit namespace or an implied namespace. An implied namespace is a directory path or OpenVMS file specification preceded by two caret characters: either "^^*dir*" (on the current system), or "^*system*^*dir*".

To understand **$ZUTIL(20)**, you need to keep the following in mind. When a job starts, it uses routines in the current namespace. After you issue **$ZUTIL(20,new)**, all subsequent routines come from *new*.

To determine the current namespace, use the $ZNSPACE special variable or $ZUTIL(5). To change the current namespace, use the ZNSPACE command or $ZUTIL(5).

If you issue **$ZUTIL(20,*new*,*second*,*third*)**, all subsequent routines located in *new* also come from *new*. If Caché does not find a routine in *new*, Caché looks for the routine in *second*. If Caché does not find the routine in *second*, Caché looks for the routine in *third*, and so on.

Note: Using routines in *second* is VERY slow! You should use *second* as an aid to program development, not as a normal running procedure.

To use *second* as a development aid, make *new* your development area and *second* as a storage area of routines that work. When you issue **ZSAVE**, the routine you save always goes into *new*, even if you loaded it from *second*. As you edit and file routines, they appear in new without affecting the library in *second*, and subsequent calls for the routines get the new version from *new*.

For % routines, refer to $ZUTIL(39).

Parameters

new

Specify the primary namespace that contains the routine dataset. If used alone, this is the namespace that will be searched for all routines. If used in conjunction with *second* and *third*, *new* will be the first namespace searched for routines.

Enclose the namespace in double quotes. The namespace you specify can be either an explicit namespace or an implied namespace (a directory path or OpenVMS file specification preceded by two caret characters (^^)).

second

Specify the second namespace to search for routines. If Caché does not find a routine in *new*, it will search for it in *second*. Specify the same as *new*.

third

Specify the third namespace to search for routines. If Caché does not find a routine in *new* or *second*, it will search for it in *third*. Specify the same as *new*.

Notes

Keep the following points in mind when you use **$ZUTIL(20)**.

- Issuing **$ZUTIL(20)** without specifying other parameters does not change the current namespace.
- The **$ZUTIL(20)** return value is the primary (default) namespace for routine access that was set before you called the function.
- Calling **$ZUTIL(20)** with parameters always clears previous settings.
- Calling **$ZUTIL(20)** with a null string as a parameter ($ZUTIL(20,"")), resets the current namespace for routine access to the default namespace.
- Calling **$ZUTIL(20,n)**, where n is an integer, issues a <FUNCTION> error.
- The command does not affect globals; the original namespace still defines access privileges for globals. Your job uses globals in *the current namespace* unless you make explicit or implicit references to globals in other namespaces. In other words, the routines come from *new* (or from *second* or *third*), while the globals come from the job's current namespace.

See Also

- ZNSPACE command
- $ZUTIL(5) Display or Switch Namespace function
- $ZUTIL(90,4) Start Up in a Specified Namespace function
- $ZUTIL(90,10) Test Whether a Namespace is Defined function
- $ZNSPACE special variable
- Configuring Namespaces in *Caché System Administration Guide*

$ZUTIL(21)

Returns the location of process-private globals or deletes all process-private globals.

```
$ZUTIL(21,flag)
$ZU(21,flag)
```

Parameters

flag	*Optional* — A boolean flag. 0 returns the location of process-private globals. 1 deletes all process-private globals. The default is 0.

Description

Use **$ZUTIL(21)** to handle the process-private globals for the current process. **$ZUTIL(21,0)** returns the location of the process-private globals database in the form *sfn^dir*. If no process-private globals are defined, this defaults to 15999^0. For further information on System File Numbers (sfn), refer to $ZUTIL(49). **$ZUTIL(21,1)** deletes all process-private globals defined for the current process. It returns 1 upon successful completion, 0 upon error. **$ZUTIL(21,1)** returns 1 even if no process-private globals were defined.

Example

```
WRITE !,$ZUTIL(21,0)," before ppgs"
SET ^||a=123
SET ^||a(1,1)="John Doe"
WRITE !,$ZUTIL(21,0)," after defining ppgs"
WRITE !,$ZUTIL(21,1)
WRITE !,$ZUTIL(21,0)," after deleting ppgs"
```

See Also

- $ZUTIL(49) — Obtain database label information function
- Variables in *Using Caché ObjectScript*

$ZUTIL(22)

Specifies the form feed or backspace control code sequence.

```
$ZUTIL(22,0,ff,bs)
$ZU(22,0,ff,bs)
```

Parameters

ff	*Optional* — The new value for the form feed control code sequence. If omitted, Caché sets the system default, as defined for Device 0.
bs	*Optional* — The new value for the backspace control code sequence. If omitted, Caché sets the system default, as defined for Device 0.

Description

This function is used to change the form feed and backspace control code sequences. Specifying a zero (0) as the second parameter is mandatory.

For example, to change the form feed control sequence, use **$CHAR** in the *ff* parameter to specify the decimal code for one or more ASCII control characters.

Parameters

ff

The new form feed control code sequence. To set the printer form feed to the control character sequence CR/FF (carriage return (ASCII 13) and form feed (ASCII 12)), specify the following:

```
DO $ZUTIL(22,0,$CHAR(13,12))
```

bs

The new backspace control code sequence. To set the backspace, you must also specify the form feed control sequence, as shown in the following example:

```
DO $ZUTIL(22,0,$CHAR(13,12),$CHAR(8))
```

Notes

To determine the system default for device control character sequences, you must determine the device type for Device 0, and the control code settings for that device type.

- To determine the device type for Device 0, go to the System Management Portal, select **[Home] > [Configuration] > [Device Settings] > [Devices]** to display the current list of devices. For Device 0, click "Edit". The device type is specified by the **Sub-Type** value.
- To determine the control code sequences for Device 0, go to the System Management Portal, select **[Home] > [Configuration] > [Device Settings] > [Device Subtypes]** to display the current list of device subtypes. Click "Edit" for the device type for Device 0 to view and edit the default Backspace and FormFeed control character sequences.

See Also

- Terminal I/O in *Caché I/O Device Guide*
- The Spool Device in *Caché I/O Device Guide*

$ZUTIL(28)

Performs collation conversion.

```
$ZUTIL(28,string,flag,len)
$ZU(28,string,flag,len)
```

Parameters

string	An expression specifying a string or numeric to be converted to a specified collation type.
flag	An integer code used to specify the desired collation type. Valid values are 0 through 9.
len	*Optional* — The truncation length in characters, specified as an integer. Truncation is performed on the collation-converted string. This option can only be used with *flag* values of 7, 8, or 9. A decimal *len* value is truncated to its integer part. A negative or nonnumeric *len* value is treated as 0.

Description

$ZUTIL(28) applies the collation type specified in *flag* to *string*. The following *flag* values are supported:

flag	*Meaning*
0	EXACT: Returns *string* unchanged. Does not convert NULLs. Corresponds to the Caché SQL %EXACT function.
1	SPACE: Appends a blank to beginning of *string*.
2	MVR: Returns *string* unchanged, except for NULLs, which are converted. Provided for MultiValue support. Corresponds to the Caché SQL %MVR function.
3	PLUS: Converts numerics and numeric strings to canonical numbers. A nonnumeric string is returned as 0.
4	MINUS: Converts numerics and numeric strings to canonical numbers and appends a minus sign. A nonnumeric string is returned as 0.
5	UPPER: Converts letters to uppercase. Corresponds to the Caché SQL %UPPER function.
6	ALPHAUP: Removes leading, trailing, and embedded blanks. Removes all punctuation characters, except commas (,) and question marks (?). Converts letters to uppercase. Corresponds to the Caché SQL %ALPHAUP function.
7	SQLUPPER: Removes trailing blanks. Converts letters to uppercase. Appends a blank to beginning of string. Corresponds to the Caché SQL %SQLUPPER function.
8	SQLSTRING: Removes trailing blanks. Appends a blank to beginning of string. Corresponds to the Caché SQL %SQLSTRING function.
9	STRING: Removes leading, trailing, and embedded blanks. Removes all punctuation characters, except commas (,). Converts letters to uppercase. Appends a blank to beginning of string. Corresponds to the Caché SQL %STRING function.

Several of these collation conversions append a blank to the string. This forces numerics and the empty string to be collated as strings.

Numerics are converted to canonical form: leading and trailing zeros are removed, as is a trailing decimal point. Multiple plus and minus signs are resolved; if the resulting sign is a plus sign, it is removed.

The MINUS collation type appends a minus sign to the supplied sign before canonical resolution. Thus the MINUS collation of a negative number is a positive number. PLUS and MINUS resolve mixed numeric strings (such as “7dwarves”) by truncating the string at the first nonnumeric character. PLUS and MINUS resolve nonnumeric strings by assigning them a value of 0.

Examples

The following example shows collation conversion of a string.

```
SET x="The quick, brown Fox? "
WRITE "flag 0:",$ZUTIL(28,x,0),":",!
WRITE "flag 1:",$ZUTIL(28,x,1),":",!
WRITE "flag 2:",$ZUTIL(28,x,2),":",!
WRITE "flag 3:",$ZUTIL(28,x,3),":",!
WRITE "flag 4:",$ZUTIL(28,x,4),":",!
WRITE "flag 5:",$ZUTIL(28,x,5),":",!
WRITE "flag 6:",$ZUTIL(28,x,6),":",!
WRITE "flag 7:",$ZUTIL(28,x,7),":",!
WRITE "flag 8:",$ZUTIL(28,x,8),":",!
WRITE "flag 9:",$ZUTIL(28,x,9),":"
```

The following example shows collation conversion of a numeric.

```
SET x=+00123.4500
WRITE "flag 0:",$ZUTIL(28,x,0),!
WRITE "flag 1:",$ZUTIL(28,x,1),!
WRITE "flag 2:",$ZUTIL(28,x,2),!
WRITE "flag 3:",$ZUTIL(28,x,3),!
WRITE "flag 4:",$ZUTIL(28,x,4),!
WRITE "flag 5:",$ZUTIL(28,x,5),!
WRITE "flag 6:",$ZUTIL(28,x,6),!
WRITE "flag 7:",$ZUTIL(28,x,7),!
WRITE "flag 8:",$ZUTIL(28,x,8),!
WRITE "flag 9:",$ZUTIL(28,x,9)
```

The following example shows collation conversion of a string, with truncation.

```
SET x="The quick brown Fox"
WRITE "flag 7:",$ZUTIL(28,x,7,6),!
WRITE "flag 8:",$ZUTIL(28,x,8,6),!
WRITE "flag 9:",$ZUTIL(28,x,9,6)
```

Note that truncation is performed *after* all collation conversion is complete, including the appending of a leading blank.

See Also

- “Strings” and “Numbers” in the “Data Types and Values” chapter of *Using Caché ObjectScript*.

$ZUTIL(39)

Specifies a search path for percent (%) routines.

```
$ZUTIL(39,sys1,sys2,sys3)
$ZU(39,sys1,sys2,sys3)
```

Parameters

sys1	Specify a namespace name to use as the search path for percent (%) routines.
sys2	*Optional* — Specify a namespace name to use as the second search path for percent (%) routines.
sys3	*Optional* — Specify a namespace name to use as the third search path for percent (%) routines.

Description

$ZUTIL(39) specifies one or more namespaces to use as a search path for percent (%) routines, and thereby establishes an alternate system manager's namespace. After issuing **$ZUTIL(39)**, when you call percent routines, Caché search for these routines in the namespace(s) specified by **$ZUTIL(39)**, rather than the default system manager's namespace.

A percent routine is a routine whose name begins with a percent sign character. This naming convention is commonly used for system routines found in the system manager's namespace and available to all users. The default system manager's namespace is %SYS.

After issuing **$ZUTIL(39)**, Caché uses up to the maximum of three path specifications to find percent routines when they are called. If Caché cannot locate a percent routine in the Caché database in *sys1*, it then searches *sys2* (if you specified this parameter.) If the percent routine is not in the Caché database in *sys2*, Caché then searches *sys3* (if you specified this parameter.)

You can use a null string as a search specification to indicate the default system manager's namespace. After resetting the search path to the default system manager's namespace, subsequent calls to **$ZUTIL(39)** returns `^^c:\InterSystems\Cache\mgr\`.

A call to **$ZUTIL(39)** with no parameters returns the current namespace to search for percent routines. A call to **$ZUTIL(39)** with any number of parameters returns the namespace name for percent routine searches that existed before that **$ZUTIL(39)** call. **$ZUTIL(39)** only returns one namespace name (*sys1*), even if you have established *sys2* and *sys3*.

Note: Your system's performance suffers significantly when Caché must load a routine from *sys2* or *sys3*. For this reason, use these for program testing only. For example, if you change a few percent routines, you may want to test them in a namespace where users are not currently running.

A typical use of the **$ZUTIL(39)** function would be to run a set of % routines that are different from the ones in the system manager's namespace. For example, you might want to run a different version of the InterSystems relational database application generator (RDBMS product). Or, you might want to develop % routines in a development environment without affecting the routines in the system manager's namespace.

Note: OpenVMS: A process can use **$ZUTIL(39)** only if its UIC matches that of the original system manager's CACHE.DAT file. Otherwise, Caché issues a <PROTECT> error.

When a process switches namespaces using **$ZUTIL(5)** or the **ZNSPACE** command, **$ZUTIL(39)** information is normally reset. The only exception to this is when the process switches from an implied namespace to an implied namespace, in which case **$ZUTIL(39)** settings are preserved. An implied namespace is a directory path or OpenVMS file specification preceded by two caret characters: "^^*dir*".

Parameters

sys1

Specify the search path for percent (%) routines. If used alone, this is the search path that will be searched for all routines. If used in conjunction with *sys2* and *sys3*, *sys1* will be the first search path searched for routines. You can specify a null string as to indicate the default system manager's namespace.

sys2

Specify the second search path for percent (%) routines. If Caché does not find a routine in *sys1*, it will search for it in *sys2*. You can specify a null string as to indicate the default system manager's namespace.

sys3

Specify the third search path for percent (%) routines. If Caché does not find a routine in *sys1* or *sys2*, it will search for it in *sys3*. You can specify a null string as to indicate the default system manager's namespace.

Example

```
WRITE !,"Initial setting: ",$ZUTIL(39)
WRITE !,$ZUTIL(39,"^^c:\InterSystems\Cache\mgr\samples\","")
     ; note use of implied namespace with ^^
WRITE !,"New setting: ",$ZUTIL(39)
```

See Also

- ZNSPACE command
- $ZUTIL(5) Display or Switch Namespace function
- $ZUTIL(20) Specify the namespace that contains the routine dataset function
- $ZUTIL(68,43) Sets clearing of global vectors for the current process function
- $ZUTIL(69,43) Sets clearing of global vectors system-wide function
- $ZNSPACE special variable
- Configuring Namespaces in *Caché System Administration Guide*

$ZUTIL(49)

Obtains database label information.

```
$ZUTIL(49,dir,flag)
$ZU(49,dir,flag)
```

Parameters

dir	Name of a directory which contains a Caché .DAT database file. A quoted string containing a fully qualified pathname.
flag	*Optional* — A flag that specifies what type of database information to return. Permitted values are: 0 = database header information; 1 = cluster lock information; 2 = file expansion time; or 3 = directory location.

Description

Use the various forms of **$ZUTIL(49)** to return label information about a database. You specify the pathname of the directory containing the database, not the name of the database file itself.

$ZUTIL(49) with no parameters returns the header information for the manager's directory. **$ZUTIL(49,*dir*,*flag*)** returns the type of information specified by *flag* from the *dir* directory. This information is returned as a string of comma-separated items. The first item of this returned string is either the system file number (sfn) or a negative number indicating that the specified directory does not exist, is not mounted, or cannot be read.

You can use the wildcard options of the **$ZSEARCH** function to determine the pathnames of .DAT files on your system.

Parameters

dir

The fully qualified pathname of the directory which contains the Caché .DAT database file. Specify *dir* as a quoted string. The *dir* parameter must be present to specify a *flag* parameter. If you omit the *dir* parameter, **$ZUTIL(49)** (with no arguments) returns the header information for the system manager's directory (for example: c:\InterSystems\Cache\mgr\). You can specify *dir* as a null string ("") to return information on the current directory (for example: c:\InterSystems\Cache\mgr\user\).

On Windows and UNIX® systems you can also use the following standard pathname symbols: a single dot (.) to specify the current directory, or a double dot (..) to specify its parent directory.

Alternately, you can specify the system file number (sfn), which is an index to gfiletab, as the *dir* parameter value for most *flag* values. This system file number can be obtained from a prior call to **$ZUTIL(49)** which specified the directory name as the *dir* parameter.

flag

A flag that specifies what type of database information to return.

Value	Description
0	Return the database header information as a comma-separated string. This is the default if no *flag* parameter is specified.
1	Return cluster lock information as a comma-separated string. Do not use this *flag* value when specifying a system file number for *dir*.
2	Return the most recent database file expansion time (in **$HOROLOG** format). This value is returned as the second item in a comma-separated string.
3	Returns the directory location as the string *sys^dir*, where *sys* is the remote system number (0 if local), and *dir* is the directory name. This *flag* value can only be used when specifying a system file number (sfn) for *dir*.

Database Header Information

$ZUTIL(49), **$ZUTIL(49,dir)** and **$ZUTIL(49,dir,0)** all return the database header information, returned as a comma-separated list of values. The following table lists the values that these forms of **$ZUTIL(49)** can return for each part of this comma-separated list:

Listed Item	Description
1	This part contains either the filename status flag (if file header info cannot be obtained) or the system file number: -3: This flag indicates that *dir* specified a filename that is too long or has invalid filename syntax. No other information is returned. -2: This flag indicates that *dir* specified a file that does not exist or cannot be read. No other information is returned. -1: This flag indicates that *dir* specified a file that exists, but is currently dismounted. No other information is returned. 255 or 15999: These numbers are too large to be actual system file numbers. They indicate that the specified directory is not mounted. No other information is returned. 15999 is returned by Caché version 5.0 and subsequent versions; 255 is returned by Caché versions before 5.0. Other numbers: The system file number (index into gfiletab) of the file. If zero (0), this indicates the System Manager's directory. These values are followed by a comma-separated list of header information items, as shown below.
2	Caché block size, in bytes. Either 2048 or 8192.
3	User Identification Code (UIC), the decimal value of the two bytes of the protection code assigned to the Caché database file. For example, A UIC of [1,4] is represented by the decimal number 260 (256 + 4). 0 if no protection. (The OpenVMS operating system assigns a UIC; At Caché version 5.1 (and subsequent), Caché security no longer sets or uses a UIC value.)
4	Current size of the Caché database file in megabytes. (1MB = 1024 * 1024 bytes)
5	Number of blocks by which Caché expands the database when it runs out of room.
6	Maximum total number of blocks that can be allocated for this database. 0 = limited only by GMaxBlks.
7	Relative block number where global directory begins, counting from 1. Cannot be changed once set.
8	Relative block number where new global pointers will begin to be allocated, counting from 1.
9	Currently unused; default is 0.
10	Currently unused; default is 0.
11	Relative block number where new global data will be added, counting from 1.
12	Freeze on database error indicator. 0 = Freeze writes on error enabled. 1 = Freeze writes on error disabled.
13	Default global collation sequence for the database. 0 through 4 are prior InterSystems products (ISM); 5 is Caché standard; 10 through 127 are for collation sequences for languages other than English. 128 through 255 are string-only collation sequences that correspond to the 0 through 127 assignments. Thus 133 is the Caché standard string-only collation sequence.
14	System file number, regardless of whether the file is currently mounted or dismounted.
15	Total number of virtual volumes in the volume set.
16 through 25	These values are for internal reference only.

An examples of database header information is as follows:

```
ZNSPACE "%SYS"
SET dir=$ZUTIL(168)  ; determine Caché directory
SET dbdir=dir_"cachelib"
WRITE $ZUTIL(49,dbdir,0)
```

which returns a string of 24 comma-separated values (such as the following), each item corresponding to the items in the table above:

```
1,8192,0,90,0,0,3,16,0,0,50,1,5,1,1,320,4228,1,1123751162,1,2,11520,62464,1,
```

This example requires that UnknownUser have assigned the %DB_CACHESYS role.

Cluster Lock Information

If you specify a flag value of 1, **$ZUTIL(49,dir,1)** returns the cluster lock information for *dir* for the database. The returned string consists of four comma-separated parts:

```
sfn,cfn,locktype,lockid
```

Listed Item	Description
sfn	The system file number (index into gfiletab) of the file. If zero (0), this indicates the System Manager's directory. Negative numbers indicate that the specified directory does not exist, is not mounted, or cannot be read (see the first item in the Database Header Information table).
cfn	Cluster File Number (CFN) if the directory is a cluster-mounted database. Otherwise (non-USECLUSTER systems) returns zero (0).
locktype	The locktype status if the directory is cluster-mounted: S=shared, X=exclusive, N=null, U=unknown. Otherwise returns zero (0).
lockid	Lock ID of the distributed lock manager if the directory is cluster-mounted. Otherwise returns zero (0).

An examples of database cluster lock information is as follows:

```
ZNSPACE "%SYS"
SET dir=$ZUTIL(168)  ; determine Caché directory
SET dbdir=dir_"cachelib"
WRITE $ZUTIL(49,dbdir,1)
```

which returns the following string of 4 comma-separated values, corresponding to the rows of the table above:

```
1,0,0,0,
```

This example requires that UnknownUser have assigned the %DB_CACHESYS role.

Note: Do not specify a system file number (sfn) for *dir* for **$ZUTIL(49,dir,1)**.

For Tru64 UNIX®, the lockid is of the form: "seqn:hndl".

File Expansion Time

If you specify a flag value of 2, **$ZUTIL(49,dir,2)** returns the date and time that the last database file expansion occurred, in **$HOROLOG** format. The returned string consists of three comma-separated parts:

```
sfn,expanddate,expandtime
```

Listed Item	Description
sfn	The system file number (index into gfiletab) of the file. If zero (0), this indicates the System Manager's directory. Negative numbers indicate that the specified directory does not exist, is not mounted, or cannot be read (see the first item in the Database Header Information table).
expanddate	The date of the last file expansion, expressed as the date half of a **$HOROLOG** value.
expandtime	The time of the last file expansion, expressed as the time half of a **$HOROLOG** value.

An examples of database expansion time information is as follows:

```
ZNSPACE "%SYS"
SET dir=$ZUTIL(168)  ; determine Caché directory
SET dbdir=dir_"cachetemp"
WRITE $ZUTIL(49,dbdir,2)
```

which returns the following string of 3 comma-separated values, corresponding to the rows of the table above:

```
2,59161,40560
```

This example requires that UnknownUser have assigned the %DB_CACHESYS role.

Directory Location

If you specify a flag value of 3, and specify a system file number for the *dir* argument, **$ZUTIL(49,dir,3)** returns the remote system number and the directory pathname. The returned string separates these two parts with a caret (^):

```
sys^dir
```

Listed Item	Description
sys	The remote system number where the specified sfn is located. If zero (0), this indicates the directory is located on the local system.
dir	The pathname of the file that corresponds to the specified sfn.

If the specified system file number refers to a nonexistent file, this form of **$ZUTIL(49)** returns -2. (See the first item in the Database Header Information table for details on negative return values.)

An examples of database location information is as follows:

```
WRITE $ZUTIL(49,2,3)
```

which returns a string such as the following. Because it is a locally mounted database, the first component in this case is zero:

```
0^c:\InterSystems\Cache\mgr\cachetemp\^%DB_CACHETEMP^8^5^3^128
```

See Also

- $VIEW function
- $ZSEARCH function
- $HOROLOG special variable

$ZUTIL(53)

Passes TCP device name to child process, or returns TCP statistics.

```
$ZUTIL(53,n)
$ZU(53,n)
```

Parameters

n	*Optional* — An integer code that specifies which TCP connection statistic to return. Supported values are 2, 3, 4, and 5.

Description

$ZUTIL(53) with no argument is called by a child process to retrieve the TCP device name assigned by the parent process.

$ZUTIL(53) with an argument returns integer counts of TCP device activity. The current device must be a connected TCP device to use these functions.

- **$ZUTIL(53,2)** returns the number of bytes that have been read from the current TCP device.
- **$ZUTIL(53,3)** returns the number of bytes that have been read from the current TCP device and clears the counter.
- **$ZUTIL(53,4)** returns the number of bytes that have been written to the current TCP device.
- **$ZUTIL(53,5)** returns the number of bytes that have been written to the current TCP device and clears the counter.

Example

```
/* In child process */
SET tcp=$ZUTIL(53)
```

See Also

- JOB command
- TCP Client/Server Communication in *Caché I/O Device Guide*

$ZUTIL(55)

Returns or changes the current language mode.

```
$ZUTIL(55,n)
```

Parameters

n	*Optional* — A numeric code specifying what language mode to set.

Description

Use this form of **$ZUTIL** to examine or change the current language mode.

To return the current state of the language mode switch without changing it, issue **$ZUTIL(55)** without the *n* parameter, as follows:

```
WRITE $ZUTIL(55)
```

To change the language mode, issue **$ZUTIL(55,*n*)**. This changes the language mode and returns the previous language mode value.

You can use $ZUTIL(55) to change the language mode at the programmer prompt. (Use the complete $ZUTIL spelling, as the $ZU abbreviation is not valid in all language modes.)

When Caché executes a routine, it automatically changes the language mode to the routine's language mode. A DO command or a $$ extrinsic function temporarily changes the language mode to that of the called routine. Caché then restores the previous language mode when the routine terminates. Therefore, you need not (and cannot) change the language mode during runtime execution of a routine.

The ZLOAD command changes the language mode to the loaded routine's language mode. However, ZLOAD does not restore the previous language mode upon termination of the routine. One use of $ZUTIL(55) is to set the language mode following a ZLOAD operation.

$ZUTIL(55) can change the process' language mode at any time. However, when executing a routine you should only do this under XECUTE, and restore before leaving the XECUTE operation. This technique is commonly used to set language mode when compiling a routine.

The object code (tokens, mcode, and so onfor a routine is generated at compile time according to the language mode in effect at that time. This language mode determines the routine's runtime behavior, except for a few cases where language mode is tested at runtime.

Any form of indirection (for example, an XECUTE command, or the @ indirection operator) is evaluated at runtime according to the current language mode. This language mode may not be the same as the language mode of the running routine.

Parameters

n

This switch determines which language will be set:

n= 0	Set language mode to Caché and return the value for the previous state of the switch. Caché is the default mode for the switch.
n= 1	Set language mode to DSM-11 and return the value for the previous state of the switch.
n= 2	Set language mode to Open M [DTM] and return the previous state of the switch.
n= 5	Set language mode to DSM for OpenVMS an return the value for the previous state of the switch.
n= 6	Set language mode to DSM-J (Japanese) for OpenVMS and return the value for the previous state of the switch.
n= 7	Set language mode to DTM-J (Japanese) for OpenVMS and return the value for the previous state of the switch.
n= 8	Set language mode to MSM and return the value for the previous state of the switch. MSM mode changes the use of the $ZC ($ZCHILD) special variable.

Notes

Keep the following points in mind when you use **$ZUTIL(55)**:

- When you are in Caché mode, you can abbreviate **$ZUTIL** to **$ZU**. You cannot use this abbreviation in other language modes.
- If you change the language mode with the **$ZUTIL(55,n)** switch so that it does not match that of the loaded routine, and then attempt to do a **ZINSERT**, you will get a <LANGUAGE MISMATCH> error message.
- In language modes 1 (DSM-11) and 8 (MSM) five-character error codes are used: <FUNCT>, <UNDEF>, <SYNTX>, <DIVER> (division error), and so forth.

Examples

```
SET rtn=$ZUTIL(55,x)

DO $ZUTIL(55,x)

SET rtn=$ZUTIL(55)
```

See Also

- XECUTE command
- ZLOAD command
- $ZCHILD special variable
- Open M Language Compatibility in *Using Caché ObjectScript*

$ZUTIL(56,2)

Locates source file and line of code for last Caché ObjectScript error.

```
$ZUTIL(56,2)
$ZU(56,2)
```

Description

Use this form of **$ZUTIL** to locate the source file and the line of the last Caché ObjectScript error that occurred.

The name of the last Caché ObjectScript error that occurred is contained in the $ZERROR special variable.

See Also

- $ZERROR special variable

$ZUTIL(56,6)

Returns the operating system error code for a sequential file error.

```
$ZUTIL(56,6)
$ZU(56,6)
```

Description

When a sequential file I/O operation fails, the operating system returns an error code to Caché. Caché, in turn, issues its own corresponding error code. **$ZUTIL(56,6)** returns the operating system error code number (in angle brackets) followed by the operating system error message. These operating system errors are specific to the operating system being used.

The name of the last Caché ObjectScript error that occurred is contained in the **$ZERROR** special variable. The source file and the line of the last Caché ObjectScript error is found in **$ZUTIL(56,2)**.

Example

The following Windows example attempts to open the non-existent file “fred”. When the OPEN times out, **$ZUTIL(56,2)** returns the Windows error code and error message:

```
OPEN "C:\fred"::3
WRITE $ZUTIL(56,6)
```

This program returns: `<2> The system cannot find the file specified.`

See Also

- $ZUTIL(56,2) Error source file and code line function
- $ZERROR special variable

$ZUTIL(62)

Performs a syntax check of command line code.

```
$ZUTIL(62,n,code)
$ZU(62,n,code)
```

Parameters

n	If *n*=1, tests a standard Caché ObjectScript command line. If *n*=2, tests a Caché ObjectScript command line with (or without) formal parameters.
code	A string expression specifying a line of Caché ObjectScript code. If the code includes quote characters, these quote characters must be doubled (for example, "WRITE ""result:"",a").

Description

$ZUTIL(62) checks the syntax of a line of Caché ObjectScript code, specified as a quoted string. If *code* contains a syntax error, **$ZUTIL(62)** returns a string with the following format:

```
nn,<SYNTAX>,text
or
nn,,text
```

Where *nn* is the character position number (counting from 0) that evoked the error, <SYNTAX> is a literal (returned in a terminal session, otherwise omitted), and *text* is an error message, such as “Invalid command”, “Invalid function name”, “Invalid expression”, “Invalid special variable”, or “Expected space”. **$ZUTIL(62)** returns the first syntax error detected; the line of code may contain additional syntax errors.

If no syntax error exists, **$ZUTIL(62)** returns the null string.

$ZUTIL(62,1)

This function can be used for general ObjectScript syntax checking, including **XECUTE** and **$XECUTE** command line arguments that do not use parameter passing.

$ZUTIL(62,2)

This function can be used for checking **XECUTE** and **$XECUTE** command line arguments that use parameter passing. It recognizes a formal parameter list at the beginning of the *code* string. **$ZUTIL(62,2)** also correctly performs syntax checking on a *code* string that does not include a formal parameter list.

Examples

In following example, **$ZUTIL(62,1)** checks the syntax of a line of code and, finding no syntax error, returns the null string:

```
SET err=$ZUTIL(62,1,"    SET x=1 HANG 5")
IF err="" { WRITE "syntax correct" }
ELSE { WRITE err }
QUIT
```

Note that leading blanks are ignored when checking syntax; you may include or omit leading blanks.

In following example, **$ZUTIL(62,1)** checks the syntax of a line of code and finds a syntax error (SET cannot use the '= operator).

```
  SET err=$ZUTIL(62,1,"    SET x'=1")
  IF err="" { WRITE "syntax correct" }
  ELSE { WRITE err }
  QUIT
```

It returns: `9,,Error in SET command`. Note that the apostrophe character is at position 9, counting from 0 (and including the four leading blanks).

The following example tests several lines of program code, and returns a syntax error for each:

```
TestLines
  SET a="WRING "         ; not a command
  SET b="WRITE $FRED "   ; not a special variable
  SET c="QUIT: "         ; postconditional missing
  SET d="HANG"           ; argument missing
  SET e="WRITE $FRED() " ; not a function
  SET f="$ZPI "          ; line must begin with command
Top
  SET num=97  ; ASCII code for "a"
  SET x=$CHAR(num)
Loop
  WHILE num<103 {
     SET err=$ZUTIL(62,1,@x)  ; Note use of indirection
     IF err="" { WRITE !,"line ",x," syntax correct" }
     ELSE { WRITE !,"line ",x," ",err }
     SET num=num+1
     SET x=$CHAR(num)
     }
  QUIT
```

This example returns the following:

```
line a 5,,Invalid command
line b 11,,Invalid special variable
line c 6,,Invalid expression
line d 5,,Expected space
line e 11,,Invalid function name
line f 0,,Invalid command
```

Note that *nn* refers to the point in the command line where the parser determines that the syntax is invalid; usually this is the first blank or other punctuation character after the identifier. However, in the last case, the first character (position 0) is tested to see if it is a valid character for a command name.

See Also

- XECUTE command
- $TEXT function
- $XECUTE function

$ZUTIL(67)

Returns information from the Process Table about a specified process.

Description

The **$ZUTIL(67)** functions are used to return information about a process from the Process Table (pidtab). InterSystems recommends that you use the SYS.Process class methods to return this information, rather than the **$ZUTIL(67)** functions.

Note: The following pages describe the legacy **$ZUTIL(67)** functions. These functions are considered legacy as of Caché 5.0, and should not be used in new programming. They are described here solely for compatibility with legacy applications:

- $ZUTIL(67,0) Return Activity State of a Specified Process

$ZUTIL(68)

Sets or clears configuration options for the current process.

```
$ZUTIL(68,subfunc,n)
$ZU(68,subfunc,n)
```

Parameters

subfunc	An integer specifying which **$ZUTIL(68)** subfunction to invoke.
n	*Optional* — The value used to set the invoked subfunction, either a boolean value, or occasionally a numeric code. If this parameter is omitted, the **$ZUTIL(68)** subfunction returns its current value.

Description

The **$ZUTIL(68)** subfunctions set or clear process flags for the current process.

Although all of the **$ZUTIL(68)** subfunctions control a different aspect of process operation, all of them have the following operating characteristics in common:

- Whenever you issue a call to **$ZUTIL(68,*subfunc*)** without specifying the *n* parameter, **$ZUTIL(68,*subfunc*)** returns the current setting of the *subfunc* switch.
- Whenever you make a call to **$ZUTIL(68,*subfunc*,*n*)**, the function sets the switch to the new value specified by *n* and displays the *previous* switch value.

Almost all of the **$ZUTIL(68)** flags can also be set system-wide using the corresponding **$ZUTIL(69)** subfunctions. The one exception is **$ZUTIL(68,25)**.

Many of these flags take a system-wide default. These defaults are configured using the System Management Portal. Select **[Home] > [Configuration] > [Compatibility Settings]**.

Subfunctions

The following table shows the **$ZUTIL(68)** subfunctions recognized by Caché. For each subfunction the meaning and default are also given.

Subfunction Number and Operation	Default Setting
$ZUTIL(68,1) Null Subscript Mode Process Switch	0
$ZUTIL(68,2) Default OPEN Mode for Sequential Files	0
$ZUTIL(68,3) Automatic Sequential File Creation Mode Process Default	0
$ZUTIL(68,5) Argumentless BREAK Process Switch	1
$ZUTIL(68,6) Reliable SET Networking Mode Process Switch	0
$ZUTIL(68,7) External Reference Mode Process Switch	0
$ZUTIL(68,11) Read Line Recall Mode Process Switch	1
$ZUTIL(68,15) Modem Disconnect Detection	0
$ZUTIL(68,21) Synchronous Transaction Commit Mode	0
$ZUTIL(68,22) $X Update Mode for Escape Sequences	platform-dependent
$ZUTIL(68,25) Dynamically Set or Remove Batch Status	0
$ZUTIL(68,26) Default Display of Current Namespace	1 (Windows only) 0 (All others)
$ZUTIL(68,27) Network Hardening for the Current Process	1
$ZUTIL(68,28) Control Root (Unsubscripted) Node Kills	0
$ZUTIL(68,30) Error-Handling Behavior (to allow use of DSM language-mode error-handling)	0
$ZUTIL(68,32) Date Range and Invalid Date Behavior	0
$ZUTIL(68,34) Process Interruptible by Asynchronous Errors	1
$ZUTIL(68,39) Disable or Enable Caching	0
$ZUTIL(68,40) End-of-File Handling for Sequential Files (to support porting of MSM routines)	0
$ZUTIL(68,42) $JOB Format for Process	0
$ZUTIL(68,43) Clearing of Global Vectors (when calling $ZUTIL(5).)	0
$ZUTIL(68,45) Truncate Numbers During String-to-Number Conversion (to support porting of MSM routines)	0
$ZUTIL(68,51) Namespace Default Directory Assignment	0
$ZUTIL(68,55) $X/$Y Behavior for TCP Devices	0
$ZUTIL(68,60) Asynchronous Telnet Disconnect Errors	0
$ZUTIL(68,63) Lowercase “e” as a Scientific Notation Exponent Symbol	1
$ZUTIL(68,66) Suppress Telnet NUL at End-of-Line	0

Subfunction Number and Operation	Default Setting
$ZUTIL(68,67) Stack and Register Usage Message Box Display	0
$ZUTIL(68,70) $DOUBLE INF and NAN Behavior	1

Note: **$ZUTIL(69)** uses many of the same subfunction numbers as **$ZUTIL(68)**. System-wide subfunction options that cannot be changed on a per-job basis cause a <FUNCTION> error if you attempt to use them with **$ZUTIL(68)**.

See Also

- $ZUTIL(69) Set System-wide Flags and Defaults functions

$ZUTIL(68,1)

Enables or disables use of null subscripts for the current process.

```
$ZUTIL(68,1,n)
$ZU(68,1,n)
```

Parameters

n	A boolean that specifies the null subscript mode setting for the current process.

Description

The **$ZUTIL(68,1)** function enables or disables a mode that allows setting and referencing globals with null subscripts within the current process.

If null subscripted globals are not enabled (*n*=0), attempting to set a null subscripted variable, such as `SET ^x("")=99`, or to reference a null subscript, such as `WRITE ^x("")`, results in a <SUBSCRIPT> error. For details on <SUBSCRIPT> error format, refer to the **$ZERROR** special variable.

If null subscripted globals are enabled (*n*=1), you can set a null subscripted variable, such as `SET ^x("")=99`, just like any other variable. Referencing an undefined null subscripted variable results in a <UNDEFINED> error.

Invoking **$ZUTIL(68,1)** without specifying *n* returns the current switch setting.

Parameters

n

A boolean switch that specifies the null subscript mode setting for the current process.

0	Disables setting/referencing null subscripted globals (the default).
1	Enables setting/referencing null subscripted globals.

Note: This feature helps you convert applications that use null subscripts. InterSystems recommends that you not use it for new development.

This system-wide default behavior can be configured using the **$ZUTIL(69,1)** function. For further details, see $ZUTIL(69,1) — Null Subscript Mode System Default.

Example

The following example tests the null subscript mode. If null subscripts are enabled for the current process, it creates some.

```
NullSubsTest
   SET b=$ZUTIL(68,1)
   IF b=1 {
      WRITE !,"null subscripts enabled locally",!
UseNullSubs
      SET ^x("")=99,^x(0)=0,^x(1)=1
      ZWRITE ^x
      QUIT  }
  ELSE {
      WRITE !,"will enable null subscripts"
      SET y=$ZUTIL(68,1,1) ; enable for current process
      WRITE !,"null subscripts now enabled locally"
      GOTO UseNullSubs
    }
    QUIT
```

See Also

- $ZUTIL(69,1) — Null Subscript Mode System Default function

$ZUTIL(68,2)

Sets sequential file open mode for the current process.

```
$ZUTIL(68,2,n)
$ZU(68,2,n)
```

Parameters

n	The boolean value that specifies the process-specific default mode for sequential files on **OPEN**.

Description

The **$ZUTIL(68,2)** function specifies the open mode for sequential files opened by the current process. When the current process issues an **OPEN** command for a sequential file, that file is opened either in read-only mode (R) or read/write mode (RW). Calling **$ZUTIL(68,2)** specifies this open mode for the current process, overriding the system-wide default setting. On OpenVMS systems, the default value is 1. On all other platforms, the default value is 0.

Invoking **$ZUTIL(68,2)** without specifying *n* returns the current switch setting.

This system-wide default behavior can be configured using the **$ZUTIL(69,2)** function. For further details, see $ZUTIL(69,2) — Default Open Mode for Sequential Files.

Parameters

n

The boolean value that specifies the default mode for sequential files on **OPEN** for the current process. 0 means read-only (R) is the default. 1 means read and write (RW) is the default.

Example

The following example sets the open mode default, then opens a sequential file:

```
SET x=$ZUTIL(68,2)
IF x=0 {
  SET y=$ZUTIL(68,2,1)
  WRITE !,"Set the open mode to:",$ZUTIL(68,2)
}
ELSE {
  WRITE !,"Open mode was already set to:",$ZUTIL(68,2)
}
OPEN "c:\myfiles\seqtest1":("NRW"):5
; ...
QUIT
```

See Also

- $ZUTIL(69,2) — Set Open Mode for Sequential Files System-wide function
- OPEN command
- Sequential File I/O in *Caché I/O Device Guide*

$ZUTIL(68,3)

Sets automatic sequential file creation option for the current process.

```
$ZUTIL(68,3,n)
$ZU(68,3,n)
```

Parameters

n	The boolean value that specifies whether a sequential file is automatically created on **OPEN**.

Description

This **$ZUTIL(68,3)** function specifies if a sequential file is created automatically in response to an **OPEN** command that specifies a file that does not already exist. In Caché, if you attempt to open a file in "W" or "RW" mode, but that file does not yet exist, the default behavior is platform-dependent. On Windows and UNIX® systems, the default is for Caché to hang until the file is actually created, or until the process is resolved by a timeout expiration or by calling the RESJOB utility. On OpenVMS systems, the default is to create a new sequential file.

This system-wide default behavior can be configured using the **$ZUTIL(69,3)** function. See $ZUTIL(69,3) — Automatic File Creation Mode System Default.

Setting **$ZUTIL(68,3)** causes Caché to create a sequential file automatically when you issue the **OPEN** command in "W" or "RW" mode for a file that does not already exist. This **$ZUTIL(68,3)** setting governs all sequential file opens in the current process.

The system-wide or current-process default behavior can be overridden for individual **OPEN** commands using the “E” (or /CREATE) mode parameter. The **OPEN** mode parameters are listed in Sequential File I/O in the *Caché I/O Device Guide*.

Invoking **$ZUTIL(68,3)** without specifying *n* returns the current switch setting.

Parameters

n

The switch that specifies whether or not a file is created when the process attempts to open a nonexistent file. 0 specifies that automatic new file creation is disabled. 1 specifies that automatic new file creation is enabled. The system default setting is 0.

Example

```
SET x=$ZUTIL(69,3),y=$ZUTIL(68,3)
WRITE !,"system-wide auto-create on open:",x
WRITE !,"current process auto-create on open:",y
IF y=0 {
  SET z=$ZUTIL(68,3,1)
  WRITE !,"Set the process auto-create option to:",$ZUTIL(68,3)
}
ELSE {
  WRITE !,"Auto-create on open was already set to:",$ZUTIL(68,3)
}
OPEN "c:\myfiles\seqtest2":("RW"):5
; ...
QUIT
```

See Also

- $ZUTIL(69,3)— Automatic File Creation Mode System Default function
- OPEN command
- Sequential File I/O in *Caché I/O Device Guide*

$ZUTIL(68,5)

Enables or disables processing of argumentless BREAK commands for the current process.

```
$ZUTIL(68,5,n)
$ZU(68,5,n)
```

Parameters

n	The boolean value that specifies whether or not argumentless **BREAK** commands are enabled for the current process.

Description

The **$ZUTIL(68,5)** function enables or disables the processing of argumentless **BREAK** commands on a per-process basis. By default, Caché does process argumentless **BREAK** commands.

InterSystems provides this capability for compatibility with non-Caché products. This switch makes it easier for you to port code from an Open M [DSM] system to other InterSystems implementations. **$ZUTIL(68,5)** simulates the behavior of the DSM commands **ZBREAK ON** and **ZBREAK OFF**.

This system-wide default behavior can be configured using the **$ZUTIL(69,5)** function. Refer to $ZUTIL(69,5) — Argumentless BREAK System Default.

Invoking **$ZUTIL(68,5)** without specifying *n* returns the current switch setting.

Parameters

n

The switch that specifies the default behavior of argumentless **BREAK**. 0 specifies that Caché treats an argumentless **BREAK** as a no-op. 1 specifies that Caché executes **BREAK** on argumentless **BREAK**. 1 is the default.

See Also

- BREAK command

- $ZUTIL(69,5) Argumentless BREAK System Default function
- Debugging chapter in *Using Caché ObjectScript*

$ZUTIL(68,6)

Enables or disables reliable SET networking mode for the current process.

```
$ZUTIL(68,6,n)
$ZU(68,6,n)
```

Parameters

n	The boolean value that specifies whether or not Reliable SET Networking mode is enabled for the current process.

Description

Use **$ZUTIL(68,6)** to enable or disable Reliable SET Networking mode for the current process.

When Reliable SET mode is enabled, each request of a **SET**, **KILL**, or **LOCK** command that results in a network error message such as <PROTECT>, <DIRECTORY>, or <FILEFULL> waits for a reply before proceeding. When Reliable SET mode is disabled, the local system ignores errors incurred by **SET**, **KILL**, and **LOCK** requests.

You can switch Reliable SET mode on and off as often as you wish, even in the middle of a stream of requests.

The system default is Reliable SET mode disabled (0). This system-wide default behavior can be configured using the **$ZUTIL(69,6)** function. Refer to $ZUTIL(69,6) — Reliable SET Networking Mode System Default.

Invoking **$ZUTIL(68,6)** without specifying *n* returns the current switch setting.

Parameters

n

The value that specifies if the Reliable SET mode is on or off. 0 disables Reliable SET mode as a default. The system default setting is 0. 1 enables Reliable SET mode as a default.

See Also

- KILL command
- LOCK command
- SET command
- $ZUTIL(69,6) Reliable SET Networking Mode System Default function

$ZUTIL(68,7)

Retains or strips extended global reference from globals returned to the current process.

```
$ZUTIL(68,7,n)
$ZU(68,7,n)
```

Parameters

n	The boolean value that specifies whether or not extended global references are retained for the current process.

Description

The **$ZUTIL(68,7)** function specifies whether **$QUERY** or **$NAME** should return an extended global reference, if the first argument of **$QUERY** or **$NAME** contains an extended global reference. A value of 0 (the default) returns the extended global reference. A value of 1 strips the extended global reference before returning the global. Invoking the **ZWRITE** command returns the extended global reference regardless of the **$ZUTIL(68,7)** setting. For further information on extended global references, see Extended Global References in *Using Caché Globals*.

Invoking **$ZUTIL(68,7)** without specifying *n* returns the current switch setting.

This function overrides the system default for the current process. The system-wide default behavior is configurable. Go to the System Management Portal, select **[Home] > [Configuration] > [Compatibility Settings]**. View and edit the current setting of **RefInKind**. The default is "false". To override this default system-wide, you can use **$ZUTIL(69,7)**.

Parameters

n

A boolean switch to set the default setting. 0 specifies to retain the extended global reference; this enables reference-in-kind as a default. 1 specifies to strip the extended global reference and return just the global; this disables reference-in-kind as a default. The system default setting is 0.

The 1 value for *n* is provided for backward compatibility.

Examples

The following example first displays the specified system global with extended global reference, then displays the system global after stripping the extended global reference:

```
DO $ZUTIL(68,7,0)
WRITE !,$QUERY(^|"%SYS"|SYS(""))
DO $ZUTIL(68,7,1)
WRITE !,$QUERY(^|"%SYS"|SYS(""))
```

returns:

```
^|"%SYS"|SYS("BACKUPDB","CACHEAUDIT")
^SYS("BACKUPDB","CACHEAUDIT")
```

See Also

- $NAME function
- $QUERY function
- $ZUTIL(69,7) Strip Extended Global References System-wide function

$ZUTIL(68,11)

Enables or disables read line recall for the current process.

```
$ZUTIL(68,11,n)
$ZU(68,11,n)
```

Parameters

n	The boolean value that specifies whether read line recall mode is enabled for the current process. 1=enabled. 0=disabled.

Description

The **$ZUTIL(68,11)** function enables or disables read line recall for terminal devices opened by the current process. Read line recall is only used by terminal devices. The current **$ZUTIL(68,11)** setting is only applied when a terminal device is explicitly opened using the **OPEN** command. The default setting is read line recall enabled.

Read line recall mode is established according to the following rules:

- The system-wide default behavior can be configured using the $ZUTIL(69,11) function.
- **$ZUTIL(68,11)** overrides this system default for terminals opened by the local process.
- The **OPEN** command can set the read line recall mode for a terminal. The **OPEN** can specify the R protocol (enable) or the N protocol (disable). If neither protocol is specified, **OPEN** takes its read line recall setting from the current value established by **$ZUTIL(68,11)**.
- The **USE** command can specify the R protocol (enable) or the N protocol (disable) to change the **OPEN** mode. If neither protocol is specified, **USE** takes its setting from the initial **OPEN** mode value (*not* from the current **$ZUTIL(68,11)** or **$ZUTIL(69,11)** default).
- An implicit open of an active device, such as issuing a **BREAK** command, reopens the device in the same mode as the initial **OPEN** command.

You cannot use **$ZUTIL(68,11)** to override an **OPEN** or **USE** setting for an active terminal. To change read line recall for an already open terminal device, you must explicitly reopen the device. You can use **$ZUTIL(68,11)** to change the process default, then issue an **OPEN 0** command, which reopens the active terminal device, applying the current process default. See Terminal I/O in *Caché I/O Device Guide* for details on using protocols.

Invoking **$ZUTIL(68,11)** without specifying *n* returns the current switch setting.

Read Line Recall

Read line recall mode provides line recall of editable lines as input for **READ** operations from a terminal. These recallable lines include both previous **READ** input lines and previous command lines. Echoing of input lines is a necessary precondition for read line recall.

Caché supports read line recall for both variable-length terminal reads (`READ var`) and fixed-length terminal reads (`READ var#n`). Caché does not support read line recall for single-character terminal reads (`READ *var`).

For a fixed-length terminal read, the recalled line is truncated to one character less than the number of characters specified in the **READ**. This final **READ** character position is reserved for typing a line termination character, specifying an edit character, or adding one more data character.

When read line recall is active, you can provide input to a **READ** by using the **Up Arrow** and **Down Arrow** keys to recall a previous terminal input line. You can then use the **Left Arrow**, **Right Arrow**, **Home**, and **End** keys to position the cursor for

editing the recalled line. You can use the **Backspace** key to delete a character, **Ctrl-X** to delete the entire line, or **Ctrl-U** to delete all of the line to the left of the cursor.

When read line recall is not active, the four **Arrow** keys, the **Home** key, and the **End** key all issue a line termination character. You can use the **Backspace** key to delete a single input character, and **Ctrl-X** (or **Ctrl-U**) to delete the entire input line. Read line recall can be deactivated by using the -R protocol, or by specifying the N, I, S, or T protocols, as described in the Terminal I/O chapter of the *Caché I/O Device Guide*.

Parameters

n

The boolean switch that controls read line recall mode.

0	Disables read line recall mode.
1	Enables read line recall mode. The system default setting is 1.

See Also

- $ZUTIL(69,11) Read Line Recall Mode System Default function
- OPEN command
- READ command
- Terminal I/O in *Caché I/O Device Guide*

$ZUTIL(68,15)

Enables or disables I/O device disconnect detection for the current process.

```
$ZUTIL(68,15,n)
$ZU(68,15,n)
```

Parameters

n	The boolean value that specifies whether Caché detects I/O disconnection.

Description

The **$ZUTIL(68,15)** function determines how Caché responds to the disconnection of the principal I/O device being accessed from the current process. When set to 1, the process receives a <DSCON> error when a disconnect is detected during a Caché ObjectScript **READ** or **WRITE** command to the device. This is known as “error on disconnect.” When set to 0, the process exits without reporting an error to the application when a disconnect is detected. The default is 0.

You should be aware that when error on disconnect is enabled, a process continues to execute after its principal device has been disconnected. It is the responsibility of the application to detect the disconnect condition and exit gracefully. Use care when enabling error on disconnect. The application must be prepared to recognize the <DSCON> error and handle it appropriately in error traps.

Error on disconnect is only applicable to TCP devices and to terminal devices where a disconnect can be recognized. Examples are modem controlled terminals and Windows Telnet, Windows LAT, and Windows local cterm (TRM:) connections. Error on disconnect is only applicable to the principal device.

You can also test the **$ZA** special variable to check for terminal disconnection. When bit 11 of **$ZA** for the principal device is set, the principal device has disconnected.

This function overrides the system-wide I/O device disconnect detection default. To set the system-wide default behavior, go to the System Management Portal, select **[Home] > [Configuration] > [Compatibility Settings]**. View and edit the current setting of **DisconnectErr**. When "true", Caché issues a <DSCON> error if a disconnect occurs. When "false", no error is issued. The default is "false". To override this default system-wide, you can use **$ZUTIL(69,15) — I/O Device Disconnect Detection System Default**.

Invoking **$ZUTIL(68,15)** without specifying *n* returns the current switch setting.

Parameters

n

The value that specifies whether Caché detects I/O device disconnection.

0	Disables disconnection detection (the default).
1	Enables disconnection detection.

See Also

- $ZUTIL(69,15) I/O Device Disconnect Detection System Default function
- $ZA special variable
- Terminal I/O in *Caché I/O Device Guide*
- TCP Client/Server Communication in *Caché I/O Device Guide*

$ZUTIL(68,21)

Sets synchronous commit mode for the current process.

```
$ZUTIL(68,21,n)
$ZU(68,21,n)
```

Parameters

n	A boolean value that specifies whether or not synchronous transaction commit mode is enabled for the current process.

Description

You can use **$ZUTIL(68,21)** to set the synchronous transaction commit mode for the current process. This enables or disables synchronizing **TCOMMIT** with the corresponding journal write operation. Every TCOMMIT command requests a flush of the journal data involved in that transaction to disk. When **$ZUTIL(68,21)** is enabled (set to 1) a **TCOMMIT** does not complete until the journal data write operation completes. When set to 0, **TCOMMIT** does not wait for the write operation to complete.

Invoking **$ZUTIL(68,21)** without specifying *n* returns the current switch setting.

$ZUTIL(68,21) can be used when synchronous transaction commit mode is turned off system-wide (usually to improve performance), but the current process requires synchronous commit — for example, when interfacing with another system.

You can use **$ZUTIL(69,21)** to set the synchronous transaction commit mode system-wide. Setting **$ZUTIL(69,21)** changes the system configuration setting and is persistent across Caché shutdowns and startups.

This system configuration setting can also be changed using the System Management Portal. Select **[Home] > [Configuration] > [Compatibility Settings]**. View and edit the current setting of **SynchCommit**. The default is "false".

Parameters

n

The boolean switch that determines the default mode for synchronous transaction commit: 0 = **TCOMMIT** completion does not wait for journal file write completion. 1 = **TCOMMIT** completion waits for the journal file write operation to complete. The default is 0.

See Also

- TCOMMIT command
- $ZUTIL(69,21) Set synchronous commit mode system-wide function

$ZUTIL(68,22)

Sets handling of escape sequences when $X is updated for the current process.

```
$ZUTIL(68,22,n)
$ZU(68,22,n)
```

Parameters

n	The numeric code that specifies the **$X** update mode for the current process. This mode should correspond to your computer platform and Intersystems software.

Description

The **$ZUTIL(68,22)** function specifies how Caché handles escape sequences when updating the **$X** special variable. When reading a string containing an escape sequence, various InterSystems products handle updating the **$X** special variable differently. You can control the way Caché updates **$X** when writing a string containing an escape sequence. Default behaviors for various implementations (including non-InterSystems ones) are:

- UNIX® parses the ANSI-standard escape sequence and counts the rest of the non-escape characters in the string against **$X**.
- Alpha OpenVMS does not count any more characters in the string against **$X** after encountering an escape character ($CHAR(27)).
- Open M [DSM] counts all characters in a string, including the escape character, against **$X**.
- MSM counts all characters in the string, except for the escape character, against **$X**.

To alter the system default setting for **$X** updating, see $ZUTIL(69,22) — $X Update Mode for Escape Sequences.

Invoking **$ZUTIL(68,22)** without specifying *n* returns the current switch setting.

Parameters

n

This argument of **$ZUTIL(68,22,n)** specifies the method of updating **$X** for an individual process:

0	Specifies UNIX® and Windows default behavior.
1	Specifies Open M [DSM] default behavior.
2	Specifies MSM default behavior.
3	Specifies Alpha/VAX default behavior.

On UNIX® and Windows systems, the default value is 0; on OpenVMS (VAX and Alpha) systems, the default value is 3.

See Also

- $ZUTIL(69,22) $X Update Mode for Escape Sequences function
- $X special variable

$ZUTIL(68,25)

Sets batch or interactive status for the current process.

```
$ZUTIL(68,25,n)
$ZU(68,25,n)
```

Parameters

n	The boolean value that specifies whether batch or interactive status is active for the current process.

Description

The **$ZUTIL(68,25,n)** function specifies either batch or interactive status for the current process. This allows you to balance resources more effectively.

When a new Caché process is created, its batch status is 0 (it is categorized as an interactive process). The following code sets its batch status to 1 (categorized as a batch process), which lessens its resource usage.

```
SET flag=$ZUTIL(68,25,1)
WRITE !,"Batch status--former:",flag," current:",$ZUTIL(68,25)
```

As with all **$ZUTIL(68)** functions, the value returned is the previous setting value, not the new setting.

Once a process has been set in batch status, it keeps that status until it terminates, or until you explicitly return it to interactive status as follows:

```
SET flag=$ZUTIL(68,25,0)
WRITE !,"Batch status--former:",flag," current:",$ZUTIL(68,25)
```

Invoking **$ZUTIL(68,25)** without specifying *n* returns the current switch setting.

Parameters

n

The boolean switch that controls the status of the issuing process:

0	Clears batch status and gives the process interactive status. The system default setting is 0.
1	Gives the process batch status.

Example

You can employ the following sequence of commands to:

- Save the current status value
- Set a potentially different status
- Call a routine or subroutine
- Restore the original status

```
BatchStatus
   SET oldflag=$ZUTIL(68,25) ; record initial status
   IF oldflag=0 {
     WRITE !,"Already in Interactive mode"
     ; call interactive routine
   }
   ELSE {
     SET flag=$ZUTIL(68,25,0)
     WRITE !,"Set to Interactive mode"
     ; call interactive routine
   }
   SET newflag=$ZUTIL(68,25,oldflag)
      ; restore initial status and return current status
   IF oldflag'=newflag {
     WRITE !,"batch status changed" }
   ELSE {
     WRITE !,"batch status unchanged" }
   QUIT
```

There is not a corresponding **$ZUTIL(69)** (system-wide) function for **$ZUTIL(68,25)**.

See Also

- JOB command

$ZUTIL(68,26)

Sets namespace display in programmer prompt for the current process.

```
$ZUTIL(68,26,n)
$ZU(68,26,n)
```

Parameters

n	The boolean value that specifies whether Caché includes the current namespace name in the terminal prompt for the current process.

Description

The **$ZUTIL(68,26)** function specifies whether the process' current namespace name appears in the programmer-mode prompt for Caché Terminal. The current namespace name is contained in the **$ZNSPACE** special variable. The current namespace name can be an explicit namespace or an implied namespace.

Issuing **$ZUTIL(68,26,1)** sets the switch for the current process, so that the namespace name is displayed. Issuing **$ZUTIL(68,26,0)** clears the switch for the current process, so that the namespace name is not displayed. To set this switch for the current process from the programmer prompt, use the **XECUTE** command, as follows:

```
USER>XECUTE "SET x=$ZUTIL(68,26,0)"

>
```

If you call **$ZUTIL(68,26)** without specifying *n*, **$ZUTIL(68,26)** returns the current switch setting.

The initial value of this switch depends on the both the system platform and the system-wide default setting. The system default setting is 1 for Windows systems, and 0 for all other platforms.

To control whether the current namespace name appears by default as part of the prompt for *all* processes on the system, use any one of the following:

- Go to the System Management Portal, select **[Home] > [Configuration] > [Compatibility Settings]**. View and edit the current setting of **NamespacePrompt**. The default is platform-dependent.
- $ZUTIL(69,26) — Enable/Disable System Namespace Display Default function.
- The **%ZSTART** user-defined system startup routine (**ZSTU** startup routines are also supported).

You can also set various values, including the current namespace, as the terminal prompt for the current process using $ZUTIL(186).

Parameters

n

The boolean value that specifies whether Caché displays the current namespace as part of the process' programmer mode prompt.

0	Specifies that the current namespace does not appear.
1	Specifies that the current namespace does appear.

Example

The following example tests the namespace name status, and sets it to display the current namespace at the programmer prompt, if necessary.

```
SET x=$ZUTIL(68,26)
IF x=0 {
 SET y=$ZUTIL(68,26,1)
 WRITE !,"Prompt changed to display namespace"
}
ELSE {
   WRITE !,"Prompt already displays namespace"
}
```

See Also

- $ZUTIL(69,26) Set Terminal Prompt Display of Current Namespace System-wide function
- $ZUTIL(186) Set Terminal Prompt Display Values for the Current Process function

- XECUTE command
- ZNSPACE command
- $ZNSPACE special variable

$ZUTIL(68,27)

Sets or clears network hardening for the current process.

```
$ZUTIL(68,27,n)
$ZU(68,27,n)
```

Parameters

n	The boolean value that specifies whether Caché enables or disables network hardening for the current process.

Description

The **$ZUTIL(68,27)** function overrides the system-wide network hardening setting for the current process.

Invoking **$ZUTIL(68,27)** without specifying *n* returns the current switch setting.

$ZUTIL(68,27,*n*) overrides any system-wide setting. Thus, you can activate network hardening for a process even if it is deactivated system-wide. You can also deactivate network hardening for a process even if it is activated system-wide. To enable or disable network hardening system-wide, set the $ZUTIL(69,27) — Enable/Disable Network Hardening System-wide function.

Caché provides network hardening to allow user jobs to continue processing in the event of network transport errors. Network hardening allows global references to remote servers to complete successfully even if network transport errors occur. When a network transport error occurs, the network transport code reestablishes connections and retransmits network requests. Where the user or application must take some action to restore communications, the client process receives a specific error.

Note: This function provide network hardening support for DCP (Distributed Cache Protocol) distributed data management. It does not provide support for ECP (Enterprise Cache Protocol) distributed data management.

DCP (Distributed Cache Protocol) with network hardening performs multiple retries of a network request. ECP (Enterprise Cache Protocol) does not support network hardening and does not perform multiple retries. ECP issues a <NETWORK> error when encountering a network transport error; the suggested programming practice is to roll back the current transaction when encountering a <NETWORK> error, then retry the transaction. At Caché version 5.0 and subsequent, the default for distributed data caching is ECP. For further details, see ECP and DCP in the *Caché Distributed Data Management Guide*.

You can determine the network hardening process state for a specified process (and other information) using a method of the SYS.Process class, as shown in the following example:

```
ZNSPACE "%SYS"
WRITE ##CLASS(SYS.Process).%OpenId($JOB).StateGet()
```

This example requires that UnknownUser have assigned the %DB_CACHESYS role.

Parameters

n

The boolean value that specifies whether Caché enables or disables network hardening for the current process:

0	Disables network hardening.
1	Enables network hardening. The system default setting is 1.

Example

The following example tests the status of network hardening for the system and the current process, and enables it for the current process, if necessary:

```
SET x=$ZUTIL(69,27)
WRITE !,"System-wide network hardening:",x
SET y=$ZUTIL(68,27)
IF y=0 {
  SET z=$ZUTIL(68,27,1)
  WRITE !,"Network hardening enabled:",$ZUTIL(68,27)
}
ELSE {
  WRITE !,"Network hardening was already enabled"
}
```

See Also

- $ZUTIL(69,27) Enable/Disable Network Hardening System-wide legacy function
- $ZUTIL(67,4) Determine the Process State legacy function

$ZUTIL(68,28)

Restricts or permits kills of root-level global nodes for the current process.

```
$ZUTIL(68,28,n)
$ZU(68,28,n)
```

Parameters

n	An integer code value that specifies whether Caché allows the current process to kill root-level (unsubscripted) global nodes. Supported values are 0, 1, and 2.

Description

The **$ZUTIL(68,28)** function specifies whether or not a process can kill root-level (unsubscripted) global nodes. By default, the switch for the process is set to zero (0). The default for the system (set through **$ZUTIL(69,28)**) is also set to zero (0). This means that killing a root-level node does not generate an error. For example, you can execute the following without an error:

```
KILL ^a
```

Invoking **$ZUTIL(68,28)** without specifying *n* returns the current switch setting.

If you set **$ZUTIL(68,28)** to disallow root-level global node kills, attempting to kill a root-level global node generates a <PROTECT> error. For details on <PROTECT> error format, refer to the **$ZERROR** special variable.

Parameters

n

An integer code that specifies whether Caché allows the processes to kill root-level global nodes: 0 = Root-level global node kills are allowed. 1 = Root-level global node kills are not allowed. 2 = Root-level global node kills are not allowed from the command line, but are allowed from routines.

Notes

$ZUTIL(68,28,*n*) overrides any system-wide setting. Thus, you can prohibit root-node deletion even if that prohibition is deactivated system-wide. You can also deactivate prohibition of root-node deletion even if that prohibition is activated system-wide.

This system-wide default behavior can be configured using the System Management Portal or the **$ZUTIL(69,28)** function. From the System Management Portal select **[Home] > [Configuration] > [Compatibility Settings].** to view the current system-wide setting of **GlobalKillEnabled**. The default is "true". Refer to $ZUTIL(69,28) — Control Root (Unsubscripted) Node Kills.

See Also

- KILL command
- $ZUTIL(69,28) Control Root (Unsubscripted) Node Kills function

$ZUTIL(68,30)

Sets error handling behavior for the current process.

```
$ZUTIL(68,30,n)
$ZU(68,30,n)
```

Parameters

n	The boolean value that specifies whether or not a process uses Caché-style error handling. 0 specifies Caché-style behavior. 1 specifies DSM-style behavior. The system default setting is 0.

Description

The **$ZUTIL(68,30)** function specifies whether a process uses Caché-style error handling (the default) or legacy (DSM) error handling.

When an error handler is invoked in Caché, that error handler remains on the stack of established error handlers. Therefore, if an error occurs while the error handler is executing, that error handler attempts to invoke itself, receives the same error again and enters an infinite loop, unless that error handler explicitly sets the **$ZTRAP** special variable to a new value. By default, processes use Caché-style error handling, so that the value of the **$ZUTIL(68,30)** switch is 0.

When an error handler is invoked in DSM, the error handler is unwound from the stack. Therefore, if an error occurs while the error handler is executing, that error is handled by the previous error handler on the stack. You can duplicate this behavior for your process by invoking this function.

Use **$ZUTIL(68,30)** to set process-specific behavior. This takes precedence over system behavior, which is set with a call to **$ZUTIL(69,30)**.

This system-wide default behavior can be configured using the **$ZUTIL(69,30)** function.

Invoking **$ZUTIL(68,30)** without specifying *n* returns the current switch setting.

Example

The following example lets you view the current state of the switch:

```
WRITE !,"error handling system-wide setting:",$ZUTIL(69,30)
WRITE !,"error handling this process setting:",$ZUTIL(68,30)
```

See Also

- $ZUTIL(69,30) Set Error Handling Behavior function
- ZQUIT command
- $ZTRAP special variable
- Error Handling in *Using Caché ObjectScript*

$ZUTIL(68,32)

Sets date range and invalid date behavior for the current process.

```
$ZUTIL(68,32,n)
$ZU(68,32,n)
```

Parameters

n	The boolean value that specifies whether or not a process uses Caché-style date behavior.

Description

The **$ZUTIL(68,32)** function performs two process-specific operations:

- It specifies how the **$ZDATE** function behaves when it receives invalid input.
- It specifies the default range of valid dates for **$ZDATE**. (This provides backward-compatibility with ISM.)

$ZUTIL(68,32) only affects the behavior of **$ZDATE**; the other date and time functions are unaffected.

This system-wide default behavior can be configured using the **$ZUTIL(69,32)** function. For further details on altering the system default for this switch, see $ZUTIL(69,32). The initial system default setting of **$ZUTIL(68,32)** is 0 for all Caché products. The initial default setting of **$ZUTIL(68,32)** is 1 for all ISM products.

Invoking **$ZUTIL(68,32)** without specifying *n* returns the current switch setting.

Parameters

n

The boolean value that specifies **$ZDATE** behavior. 0 specifies Caché-style behavior; 1 specifies ISM-style behavior. The system default setting is 0.

Notes

Calling $ZUTIL(68,32)

When you create a new process, its **$ZDATE** behavior is initialized from the current setting of **$ZUTIL(69,32)**. Calling **$ZUTIL(68,32)** overrides this behavior for the current process. Changing the setting of **$ZUTIL(69,32)** affects only processes subsequently created, not existing processes.

Setting the Behavior for Invalid Dates

An invalid date is either not a positive integer, or is not a value within the range of valid dates, as specified above.

$ZUTIL(68,32,0) causes **$ZDATE** to issue the error message <ILLEGAL VALUE> or <VALUE OUT OF RANGE> if you submit an invalid date. The behavior can be overridden by supplying an *erropt* parameter to the **$ZDATE** call.

$ZUTIL(68,32,1) causes **$ZDATE** to return the null string if you submit an invalid date. This behavior is set for any **$ZDATE** function call, regardless of the number of parameters.

Setting the Range of Valid Dates

$ZUTIL(68,32,0) sets the default range of valid dates for Caché . This range is from 0 through 2980013, inclusive, which corresponds to dates from 12/31/1840 through 12/31/9999. This range can be restricted by setting the **$ZDATE** *mindate* and *maxdate* parameters.

$ZUTIL(68,32,1) sets the default range of valid dates for ISM compatibility. This range is from 1 through 94232, inclusive, which corresponds to dates from 01/01/1841 through 12/30/2098. This date range is set for any **$ZDATE** function call which has three or fewer parameters. If a **$ZDATE** function call has more than three parameters, the valid date range is taken either from the **$ZDATE** *mindate* and *maxdate* parameters (if specified) or from the date range established for the current locale.

See Also

- $ZDATE function
- $ZUTIL(69,32) Set Date Range and Invalid Date Behavior function
- More information on locales in the article System Classes for National Language Support

$ZUTIL(68,34)

Sets whether asynchronous errors can interrupt the current process.

```
$ZUTIL(68,34,n)
$ZU(68,34,n)
```

Parameters

n	The boolean value that specifies whether or not the current process should receive asynchronous errors.

Description

The **$ZUTIL(68,34)** function specifies whether asynchronous errors can interrupt the current process.

This system-wide default behavior can be configured using the **$ZUTIL(69,34)** function. For further details, see $ZUTIL(69,34).

Invoking **$ZUTIL(68,34)** without specifying *n* returns the current switch setting.

Parameters

n

The value that specifies the process-specific behavior in response to asynchronous errors. 1 enables the reception of asynchronous errors; 0 disables the reception of asynchronous errors. The system default setting is 1.

Notes

Even if a process has disabled reception of asynchronous errors, an asynchronous error will be triggered the next time the process issues a **ZSYNC** command.

See Also

- ZSYNC command
- $ZUTIL(69,34) Processes Interruptible by Asynchronous Errors function

$ZUTIL(68,39)

Enables or disables caching for the current process.

```
$ZUTIL(68,39,n)
$ZU(68,39,n)
```

Parameters

n	The boolean value that specifies whether or not a process has caching enabled. Note that settings are: 0 = caching enabled. 1 = caching disabled.

Description

The **$ZUTIL(68,39)** function disables or enables caching for the current process. When caching is enabled, Caché will cache DCP data for quicker retrieval.

This function overrides the system-wide enable caching default.

Invoking **$ZUTIL(68,39)** without specifying *n* returns the current switch setting.

Parameters

n

The boolean switch that determines whether caching is disabled for the current process. 0 = caching enabled. 1 = caching disabled. **$ZUTIL(69,39)** uses the same values to override the default setting system-wide. Note that the boolean values used in **$ZUTIL(68,39)** and **$ZUTIL(69,39)** are the *opposite* of those found in System Configuration.

Caching enabled is the system-wide default setting. Setting **$ZUTIL(68,39)** overrides the system-wide setting for the current process.

This system-wide default behavior can be configured using the **$ZUTIL(69,39)** function.

See Also

- $ZUTIL(69,39) Disable or Enable Caching System-wide function

$ZUTIL(68,40)

Sets sequential file end-of-file handling for the current process.

```
$ZUTIL(68,40,n)
$ZU(68,40,n)
```

Parameters

n	The boolean value that sets end-of-file handling for the current process. 0=Caché default format. 1=end-of-file flagging format.

Description

The **$ZUTIL(68,40)** function specifies end-of-file flagging format. This function is provided to support certain features of Ensemble, and the porting of MSM routines that use the MSM **$ZC** function to check for end of file on sequential file reads. This MSM function should not be confused with the Caché **$ZC** function (which is an abbreviation for **$ZCHILD**).

Calling **$ZUTIL(68,40,1)** eliminates the <ENDOFFILE> error for sequential files for the current process. Instead, when the end of a file is reached, the **READ** command returns a null string, the **$ZPOS** special variable is set to "" (the null string), and the **$ZEOF** special variable is set to -1.

Invoking **$ZUTIL(68,40)** without specifying *n* returns the current switch setting.

This function overrides the system-wide end-of-file handling default for the current process.

This system-wide default behavior can be configured using the **$ZUTIL(69,40)** function. See $ZUTIL(69,40).

Parameters

n

The value that specifies how ends of files are flagged. 0 specifies that ends of files are flagged in standard Caché format (the default). 1 specifies that ends of files are flagged in **$ZPOS** and **$ZEOF**.

See Also

- $ZUTIL(69,40) End-of-File Handling for Sequential Files function
- READ command
- $ZEOF special variable
- $ZPOS special variable
- Sequential File I/O in *Caché I/O Device Guide*

$ZUTIL(68,42)

Sets $JOB format for the current process.

```
$ZUTIL(68,42,n)
$ZU(68,42,n)
```

Parameters

n	The boolean value that specifies whether or not a process uses the standard **$JOB** return string.

Description

The **$ZUTIL(68,42)** function specifies the format that the **$JOB** special variable returns for the current process.

This function provides for a unique value for **$JOB** on networked systems. This is useful when using **$JOB** to index globals accessed by more than one networked system. (In the form **$ZUTIL(68,42,1)**, the **$JOB** special variable returns a string of the format *processid:nodename*.)

This system-wide default behavior can be configured using the **$ZUTIL(69,42)** function.

Invoking **$ZUTIL(68,42)** without specifying *n* returns the current switch setting.

Parameters

n

The value that specifies the string format that **$JOB** returns. 0 specifies the standard **$JOB** format (the default); 1 specifies the format as *processid:nodename*.

Example

The following example shows the two possible values for **$JOB**:

```
SET x=$ZUTIL(68,42)
IF x=0 {
  WRITE !,"Initially standard format ",x
  WRITE !,"Standard $JOB format: ",$JOB
  SET y=$ZUTIL(68,42,1)
  WRITE !,"Extended $JOB format: ",$JOB
  SET y=$ZUTIL(68,42,0) ; restore setting
  QUIT
  }
ELSE {
  WRITE !,"Initially extended format",x
  WRITE !,"Extended $JOB format: ",$JOB
  SET y=$ZUTIL(68,42,0)
  WRITE !,"Standard $JOB format: ",$JOB
  SET y=$ZUTIL(68,42,1)  ; restore setting
  QUIT
  }
```

See Also

- $ZUTIL(69,42) $JOB Format System Default function
- $JOB special variable

$ZUTIL(68,43)

Sets clearing of global vectors for the current process.

```
$ZUTIL(68,43,n)
$ZU(68,43,n)
```

Parameters

n	The boolean value that specifies whether or not **$ZUTIL(5)** clears global vectors for the current process.

Description

The **$ZUTIL(68,43)** function specifies whether or not issuing the **$ZUTIL(5)** function with the current namespace clears global vectors for the current process.

$ZUTIL(68,43,1) provides compatibility with **$ZUTIL(5)** behavior before Caché version 3.1; this call enables the clearing of global vectors for the current process when **$ZUTIL(5)** is issued with the current namespace.

This system-wide default behavior can be configured using the **$ZUTIL(69,43)** function. For further details, see $ZUTIL(69,43).

Invoking **$ZUTIL(68,43)** without specifying *n* returns the current switch setting.

Parameters

n

The value that specifies whether or not there is global vector clearing. 0 (default Caché behavior) disables global vector clearing. 1 (legacy behavior) clears global vectors. The system default is 0.

See Also

- $ZUTIL(5) Display or Switch Namespace function
- $ZUTIL(69,43) Sets Clearing of Global Vectors System-wide function

$ZUTIL(68,45)

Sets truncation mode for string-to-number conversions for the current process.

```
$ZUTIL(68,45,n)
$ZU(68,45,n)
```

Parameters

n	The boolean value that specifies whether or not the current process truncates numbers.

Description

The **$ZUTIL(68,45)** function specifies whether or not the current process truncates very large numbers during string-to-number conversions. The default behavior is *not* to truncate numbers.

Normally, when Caché encounters a number larger than 922337203685477580E127 (or smaller than -922337203685477580E127) it converts the number to an IEEE double-precision number. This conversion only occurs during string-to-number conversions. A <MAXNUMBER> error can still occur during arithmetic operations. This conversion extends the numeric range of available numbers, but reduces the numeric precision by roughly 3 decimal digits. When Caché encounters a number too large to be represented by an IEEE double-precision number, it either issues a <MAXNUMBER> error or returns INF, depending on the setting of **$ZUTIL(68,70)** or **$ZUTIL(69,70)**. For further details on IEEE double-precision numbers, refer to the $DOUBLE function.

When **$ZUTIL(68,45,1)** is set, this conversion to an IEEE double-precision number does not occur. Instead, the number's scientific notation exponent is truncated to the biggest valid exponent value for a Caché floating point number, and the string-to-number conversion continues.

This function is provided for MSM compatibility. Specifying **$ZUTIL(55,8)** to set MSM language mode automatically sets **$ZUTIL(68,45)** and **$ZUTIL(69,45)**. (**$ZUTIL(68,45,0)** is the default setting for the switch in all language modes except MSM mode.)

This system-wide default behavior can be configured using the **$ZUTIL(69,45)** function.

Invoking **$ZUTIL(68,45)** without specifying *n* returns the current switch setting.

Parameters

n

The boolean value that specifies whether number truncation occurs. 0 preserves large numbers and performs IEEE double-precision conversion on these numbers when necessary (the default); 1 truncates large numbers during the string-to-number conversion and suppresses IEEE conversion and <MAXNUMBER> errors.

Examples

The following example demonstrates the conversion of a large number to an IEEE double-precision number when **$ZUTIL(68,45,0)** is set.

```
SET rtn=$ZUTIL(68,45,0)
WRITE !,"E127 no $DOUBLE conversion"
WRITE !,92233720368547758O7E127
WRITE !,$DOUBLE(92233720368547758O7E127)
WRITE !,"E128 automatic $DOUBLE conversion"
WRITE !,92233720368547758O7E128
WRITE !,$DOUBLE("92233720368547758O7E128")
```

The following example demonstrates the scientific notation exponent truncation of a large floating-point number when **$ZUTIL(68,45,1)** is set.

```
SET rtn=$ZUTIL(68,45,1)
WRITE !,"E127 no truncation"
WRITE !,92233720368547758O7E127
WRITE !,"E128 automatic truncation to E127"
WRITE !,92233720368547758O7E128
WRITE !,"E128 explicit $DOUBLE conversion"
WRITE !,$DOUBLE(92233720368547758O7E128)
```

See Also

- $DOUBLE function
- $ZUTIL(55) Language Mode Switch function
- $ZUTIL(69,45) Truncate Numbers During String-to-Number Conversion function
- $ZUTIL(68,70) — Set $DOUBLE INF and NAN behavior function
- $ZUTIL(69,70) — Set $DOUBLE INF and NAN behavior system-wide function

$ZUTIL(68,51)

Sets whether or not changing namespaces changes operating system directories for the current process.

```
$ZUTIL(68,51,n)
$ZU(68,51,n)
```

Parameters

n	The boolean value that specifies whether or not changing the namespace also changes the default directory for the current process.

Description

The **$ZUTIL(68,51)** function specifies — for the current process — whether changing the namespace also changes the default directory. By default, changing to a new namespace causes Caché to change the directory at the operating system level to the default directory for that namespace.

Invoking **$ZUTIL(68,51)** without specifying *n* returns the current switch setting.

To set this behavior on a system-wide basis, use **$ZUTIL(69,51)**. The system-wide default behavior is configurable. Go to the System Management Portal, select **[Home] > [Configuration] > [Compatibility Settings]**. View and edit the current setting of **SwitchOSdir**. The default is "false".

Call **$ZUTIL(168)** to explicitly set the default directory at the operating system level.

For information on mapping database directories to namespaces, refer to Configuring Caché in *Caché System Administration Guide*.

Parameters

n

The value that specifies the default behavior when changing namespaces. 0 specifies that changing the namespace automatically changes the default directory to the directory for that namespace (the default); 1 specifies that changing the namespace does not change the default directory.

See Also

- $ZUTIL(69,51) Namespace Default Directory Assignment function
- $ZUTIL(168) Set Current Working Directory function

$ZUTIL(68,55)

Selects $X/$Y behavior for TCP devices for the current process.

```
$ZUTIL(68,55,n)
$ZU(68,55,n)
```

Parameters

n	A boolean value that specifies the behavior for $X and $Y for TCP devices for the current process: 0 = Caché sets $X and $Y to conventional values (terminal emulation). 1 = Caché sets $X and $Y to specialized TCP values used for TCP data transfer output. The default is 0.

Description

You can use **$ZUTIL(68,55)** to enable or disable a specialized use of the $X and $Y special variables for TCP devices for the current process. You can use **$ZUTIL(69,55)** to set the system-wide default for this behavior.

By default, Caché sets $X and $Y to logical cursor values for all devices, with $X containing the character count for the current line and $Y containing the line count. This setting is useful when using TCP for terminal emulation. However, when using TCP strictly for data transfer output, you can set this option so that $X contains the byte count in the output buffer and $Y contains the transmitted buffer count. Use **$ZUTIL(68,55,1)** to enable this specialized $X/$Y behavior for TCP output devices opened by the current process as needed. Specialized $X/$Y behavior is not used for TCP input.

The system default setting, in which $X/$Y are set to logical cursor values for TCP terminal emulation does impose a performance penalty on TCP output. For this reason, you may wish to reset this option when using TCP devices for high-speed data transfer output. For example, Caché Server Pages (CSP) does not require $X/$Y support; CSP performance can be improved by calling **$ZUTIL(68,55,1)** to disable default $X/$Y behavior.

For further details on $X/$Y usage with TCP devices, refer to /TCPNOXY and /XYTABLE in the “OPEN and USE Command Keywords for TCP Devices” section of the TCP Client/Server Communication chapter of the *Caché I/O Device Guide*.

The setting of **$ZUTIL(68,55)** overrides the system default $X/$Y behavior when a TCP device is opened by the current process. It has no effect on a currently open TCP device. The setting of this option has no effect on $X/$Y behavior of non-TCP devices.

Invoking **$ZUTIL(68,55)** without specifying *n* returns the current switch setting.

See Also

- $ZUTIL(69,55) Select $X/$Y Behavior for TCP Devices System-wide function
- $X special variable
- $Y special variable
- TCP Client/Server Communication in *Caché I/O Device Guide*

$ZUTIL(68,60)

Sets handling of asynchronous Telnet disconnect errors for the current process.

```
$ZUTIL(68,60,n)
$ZU(68,60,n)
```

Parameters

n	A boolean value that specifies whether or not the current process should receive Telnet disconnect errors asynchronously.

Description

You can use **$ZUTIL(68,60)** to set asynchronous disconnect error handling for the current process. **$ZUTIL(68,60)** is only applicable to Telnet connections on Windows. It has no effect on any other device type or operating system.

You can use **$ZUTIL(69,60)** to set the system-wide default for this behavior.

For **$ZUTIL(68,60)** to be operational, $ZUTIL(68,15) must set to 1 for the current process.

The **$ZUTIL(68,60)** default setting is 0 (disabled). Setting **$ZUTIL(68,60,1)** overrides this default for the current process.

Invoking **$ZUTIL(68,60)** without specifying *n* returns the current switch setting.

Parameters

n

The boolean switch that determines the system-wide default mode for asynchronous disconnect errors: 0 = the process receives a <DSCON> error at the next **READ** or **WRITE** command. 1 = The process receives an asynchronous <DSCON> error immediately when a disconnect occurs on the device. This error occurs at the next command executed. **HANG** commands will be interrupted.

See Also

- $ZUTIL(69,60) — Set Handling of Asynchronous Telnet Disconnect Errors System-wide function
- $ZUTIL(68,15) — Modem Disconnect Detection for the Current Process function

$ZUTIL(68,63)

Enables or disables the use of “e” as scientific notation symbol for the current process.

```
$ZUTIL(68,63,n)
$ZU(68,63,n)
```

Parameters

n	The boolean value that specifies whether or not a process should treat lowercase “e” as a scientific notation base-10 exponent symbol. 1 = evaluate “e” as an exponent symbol. 0 = do not evaluate “e” as an exponent symbol. The default is 1.

Description

The **$ZUTIL(68,63)** function disables or enables the evaluation of the lowercase letter “e” as a scientific notation exponent symbol for the current process. **$ZUTIL(68,63)** has no effect on the evaluation of uppercase “E” as an exponent symbol.

Invoking **$ZUTIL(68,63)** without specifying *n* returns the current switch setting.

This function overrides the system-wide default, which is initialized to evaluate both “E” and “e” as exponent symbols. To set the system-wide default, you can use **$ZUTIL(69,63)**. For further details on altering the system default for this switch, see $ZUTIL(69,63).

See Also

- $ZUTIL(69,63) Disable or Enable “e” as Scientific Notation Symbol System-wide function

$ZUTIL(68,66)

Suppress Telnet NUL at end-of-line for the current process.

```
$ZUTIL(68,66,n)
$ZU(68,66,n)
```

Parameters

n	The boolean value that specifies whether or not Caché should suppress NUL following a CR at end-of-line. 1 = suppress NUL. 0 = do not suppress NUL. The default is 0.

Description

The **$ZUTIL(68,66)** function disables or enables the issuance of a NUL character (ASCII 0) following a CR character (ASCII 13) at end-of-line during Telnet transmission. On output, a Telnet network virtual terminal (NVT) performs the following default end-of-line behavior: either issues a CR (carriage return character) followed by a LF (linefeed character), or issues a CR (carriage return character) followed by a NUL character (if no LF is issued). **$ZUTIL(68,66)** can be used to suppress this NUL character.

The default is to issue both a CR and a NUL. You can use **$ZUTIL(68,66)** to reset this default for the current process. For details on overriding this default system-wide, see $ZUTIL(69,66).

$ZUTIL(68,66) affects output only. On input, the NUL character is dropped by default.

The **$ZUTIL(68,66)** setting at the time a Telnet device is opened determines the behavior of the Telnet device. Changing the **$ZUTIL(68,66)** setting has no effect on currently open Telnet devices.

$ZUTIL(68,66) is specific to Windows platforms where the Telnet protocol is implemented internally by Caché. This function has no effect on UNIX® or OpenVMS platforms.

Invoking **$ZUTIL(68,66)** without specifying *n* returns the current switch setting.

See Also

- $ZUTIL(69,66) Suppress Telnet NUL at End-of-line System-wide function

$ZUTIL(68,67)

Suppresses or displays the stack and register usage message box for the current process.

```
$ZUTIL(68,67,n)
$ZU(68,67,n)
```

Parameters

n	The boolean value that specifies whether or not Caché should display or suppress the Windows stack and register usage box for a cache.exe process when it encounters an exception. 1 = enable display of the message box. 0 = suppress display of the message box. The default is 0.

Description

The **$ZUTIL(68,67)** function disables or enables the display of a Windows message box. When enabled, this box displays stack and register usage information when a cache.exe process encounters an exception. Because processing cannot proceed until the user responds to this message box, **$ZUTIL(68,67)** provides the option to suppress the display of this message box to facilitate unattended system operation, such as scripted shutdown and restart. The default is to suppress the message box.

$ZUTIL(68,67) sets this behavior for the current process. You can set this behavior on a system-wide basis by calling **$ZUTIL(69,67)**. Invoking **$ZUTIL(68,67)** without specifying *n* returns the current switch setting.

See Also

- $ZUTIL(69,67) Suppress Message Box Display System-wide function

$ZUTIL(68,70)

Enables or disables $DOUBLE returning INF and NAN values.

```
$ZUTIL(68,70,n)
$ZU(68,70,n)
```

Parameters

n	A boolean that specifies whether to generate Caché error messages or return INF, –INF, and NAN values for unresolvable IEEE floating point conversions.

Description

The **$ZUTIL(68,70)** function sets the **$DOUBLE** function return value behavior for the current process. If 0, **$DOUBLE** returns INF (infinity), –INF, and NAN (Not A Number) for unresolvable IEEE floating point conversions. If 1, **$DOUBLE** generates Caché errors for unresolvable IEEE floating point conversions.

$ZUTIL(68,70) controls the issuing of INF, –INF, and NAN when a **$DOUBLE** numeric operation cannot be resolved to a numeric value. It does not control the issuing of INF, –INF, and NAN in all cases. **$DOUBLE** always returns INF, –INF, or NAN when you supply one of these strings as the input value, regardless of the **$ZUTIL(68,70)** setting.

Mathematical operations on **$DOUBLE** numbers that result in an INF, –INF, or NAN are controlled by **$ZUTIL(68,70)**. These include arithmetic operations, exponentiation, and logarithmic and trigonometric functions.

Invoking **$ZUTIL(68,70)** without specifying *n* returns the current switch setting.

This default behavior can be configured system-wide using the **$ZUTIL(69,70)** function. For further details, see $ZUTIL(69,70) — Set $DOUBLE INF and NAN behavior system-wide.

Parameters

n

A boolean switch that specifies the **$DOUBLE** mode setting for the current process.

0	**$DOUBLE** returns INF, –INF, or NAN when given an unresolvable numeric expression.
1	**$DOUBLE** generates Caché <MAXNUMBER>, <ILLEGAL VALUE>, and <DIVIDE> errors when given an unresolvable numeric expression. This is the default.

See Also

- $DOUBLE function
- $ZUTIL(69,70) — Set $DOUBLE INF and NAN behavior system-wide function

$ZUTIL(68,71)

Sets IP address format for the current process.

```
$ZUTIL(68,71,n)
$ZU(68,71,n)
```

Parameter

n	A boolean value to set IP address format. 0=IPv6 format disabled (IPv4 only). 1=IPv6 format enabled (both IPv4 and IPv6 supported).

Description

$ZUTIL(68,71) sets the types of IP address formats supported by the current process. IP addresses in IPv6 format can be used in TCP/IP connections. When IPv6 is enabled, Caché accepts an IP address on the **OPEN** command in either IPv6 or IPv4 format, or as a host name, with or without domain qualifiers. Caché first checks for an IPv4 format address, then for an IPv6 format address; it accepts the first successful connection.

To handle IPv6 addresses, you must first enable handling of this format by invoking either **$ZUTIL(68,71,1)** for the current process, or **$ZUTIL(69,71,1)** for all subsequent processes system-wide. The default is IPv4 format only. Further details on IPv4 and IPv6 formats can be found in the "Use of IPv6 Addressing" section of the *InterSystems Product Miscellany* article.

$ZUTIL(69,71) can be used to configure the system-wide default behavior. **$ZUTIL(68,71)** can be used to override the system setting (or default) for the local process. This configuration setting cannot be changed using the System Management Portal.

Invoking **$ZUTIL(68,71)** without specifying *n* returns the current switch setting.

See Also

- $ZUTIL(69,71) function

- JOB command
- OPEN command
- TCP Client/Server Communication in *Caché I/O Device Guide*

$ZUTIL(68,72)

Sets MVBasic handling of undefined variables.

```
$ZUTIL(68,72,n)
$ZU(68,72,n)
```

Parameters

n	The boolean value that specifies whether or not Caché should issue an error when the current MVBasic routine references an undefined variable. 1 = substitute the empty string for an undefined variable. 0 = issue an <UNDEFINED> error for an undefined variable. The default is 0.

Description

The **$ZUTIL(68,72)** function defines MVBasic behavior for the current process when it encounters a reference to an undefined variable. By default, if the current MVBasic routine references an undefined variable, Caché generates an <UNDEFINED> error. You can change this default behavior to have Caché substitute an empty string for an undefined variable, without signalling an error.

You can use $ZUTIL(69,72) to set this behavior on a system wide basis.

Invoking **$ZUTIL(68,72)** without specifying *n* returns the current switch setting.

See Also

- $ZUTIL(18) function
- $ZUTIL(69,72) function

$ZUTIL(69)

Sets system-wide configuration defaults.

```
$ZUTIL(69,subfunc,n)
$ZU(69,subfunc,n)
```

Parameters

subfunc	An integer specifying which **$ZUTIL(69)** subfunction to invoke.
n	*Optional* — The value used to set the invoked function, usually a boolean (0 or 1), occasionally a numeric code. If this parameter is omitted, the **$ZUTIL(69)** subfunction returns its current value.

Description

Use the **$ZUTIL(69)** subfunctions to set or clear system-wide configuration settings.

Although each of the **$ZUTIL(69,*subfunc*,*n*)** functions control a different aspect of system operation, all of them have the following operating characteristics in common:

- You must have the **%Admin_Manage:Use** privilege to execute any of the **$ZUTIL(69)** functions. These functions can also be invoked by % routines located in the manager's directory. For further details, refer to the Privileges and Permissions chapter of the *Caché Security Administration Guide*.
- Whenever you issue a call to **$ZUTIL(69,*subfunc*)** without specifying the third, *n* parameter, **$ZUTIL(69,*subfunc*)** returns the current setting of the switch specified by *subfunc*.
- Whenever you make a call to **$ZUTIL(69,*subfunc*,*n*)**, the function first sets the switch to the new value specified by *n*, and then returns the *previous* switch value.
- Whenever you change a system setting through the use of **$ZUTIL(69,*subfunc*,*n*)**, only processes that log in subsequent to the change are affected. Processes that logged in before the change still operate under the old switch value.

In a most cases, the **$ZUTIL(69)** function provides a system-wide override for a system-wide default value. Setting one of these **$ZUTIL(69)** functions overrides the system configuration default; it does not change the default setting. The value set by **$ZUTIL(69)** applies to all subsequent processes running on the current instance of Caché; stopping and restarting Caché causes these configuration values to revert to their defaults.

In a few cases, the **$ZUTIL(69)** function provides a means to change the system-wide default value. These defaults can also be configured using the System Management Portal. Select **[Home] > [Configuration] > [Compatibility Settings]**. In these cases, the **$ZUTIL(69)** function changes the configuration setting for the current instance of Caché, and this change persists when Caché is stopped and restarted.

The permanence of changes made using a **$ZUTIL(69)** function is described for each individual **$ZUTIL(69)** function.

Many of the **$ZUTIL(69)** flags can be set on a per-process basis by using the corresponding **$ZUTIL(68)** subfunction.

The following table shows the **$ZUTIL(69)** subfunctions recognized by Caché. For each subfunction the meaning and default are also given.

Subfunctions

Subfunction Number and Operation	Default Setting	Settable from Management Portal?	Persistent Over Caché Shutdown?
$ZUTIL(69,0) Default $ZUTIL(18) Values	0	Yes	No
$ZUTIL(69,1) Null Subscript Mode System Default	0	No	No
$ZUTIL(69,2) Default Mode for Sequential Files	0	No	No
$ZUTIL(69,3) Automatic File Creation Mode System Default	0	No	No
$ZUTIL(69,5) Argumentless BREAK System Default	1	No	No
$ZUTIL(69,6) Reliable SET Networking Mode System Default	0	No	No
$ZUTIL(69,7) Reference-in-Kind Mode System Default	0	Yes	No
$ZUTIL(69,8) ZA and ZD Modes	0	No	No

Subfunction Number and Operation	Default Setting	Settable from Management Portal?	Persistent Over Caché Shutdown?
$ZUTIL(69,10) Freeze System if Journal Full System Default	0	Yes	Yes
$ZUTIL(69,11) Read Line Recall Mode System Default	1	No	No
$ZUTIL(69,13) Asynchronous SET/KILL Errors to mneterr.log File	0	No	No
$ZUTIL(69,14) Asynchronous SET/KILL Errors to Operator Console	0	No	No
$ZUTIL(69,15) Modem Disconnect Detection System Default	0	Yes	Yes
$ZUTIL(69,19) DDP Password Security	0	No	No
$ZUTIL(69,20) Transfer Nodes with Null Subscripts with DSM-DDP	0	No	No
$ZUTIL(69,21) Synchronous Transaction Commit Mode	0	Yes	Yes
$ZUTIL(69,22) $X Update Mode for Escape Sequences	0	No	No
$ZUTIL(69,24) $ZF Process Deletion by STOP/ ID (OpenVMS)	0	No	No
$ZUTIL(69,26) System Namespace Display Default	1	Yes	Yes
$ZUTIL(69,27) Network Hardening	1	No	No
$ZUTIL(69,28) Control Root (Unsubscripted) Node Kills	0	No	No
$ZUTIL(69,30) Error Handling Behavior	0	No	No
$ZUTIL(69,31) Network Locks Handling Following a DCP Outage	0	No	No
$ZUTIL(69,32) Date Range and Invalid Date Behavior	0	No	No
$ZUTIL(69,34) Processes Interruptible by Asynchronous Errors	1	No	No
$ZUTIL(69,35) Silent Retry for domainspace Connection Attempts	0	No	No
$ZUTIL(69,37) Physical Cursor Positioning Mode	0	Yes	Yes
$ZUTIL(69,39) Caching for Future Processes	0	No	No
$ZUTIL(69,40) End-of-File Handling for Sequential Files	0	No	No
$ZUTIL(69,42) $JOB Format System Default	0	No	No
$ZUTIL(69,43) Clearing of Global Vectors	0	No	No
$ZUTIL(69,44) Nagle Algorithm for Telnet Transmissions	0	Yes	Yes
$ZUTIL(69,45) Truncate Numbers During String-to-Number Conversion	0	No	No
$ZUTIL(69,49) Logging of Transaction Rollbacks	0	No	No

Subfunction Number and Operation	Default Setting	Settable from Management Portal?	Persistent Over Caché Shutdown?
$ZUTIL(69,51) Namespace Default Directory Assignment	0	Yes	Yes
$ZUTIL(69,55) $X/$Y Behavior for TCP Devices	0	No	No
$ZUTIL(69,60) Asynchronous Telnet Disconnect Errors	0	No	No
$ZUTIL(69,63) Lowercase “e” as Scientific Notation Symbol	1	No	No
$ZUTIL(69,66) Suppress Telnet NUL at End-of-Line	0	No	No
$ZUTIL(69,67) Stack and Register Usage Message Box Display	0	No	No
$ZUTIL(69,68) Encryption of Journal Files	0	No	No
$ZUTIL(69,69) Long String Support	0	Yes	No
$ZUTIL(69,70) $DOUBLE INF and NAN Behavior	1	No	No

See Also

- $ZUTIL(68) Set Configuration Options for the Current Process functions

$ZUTIL(69,0)

Sets undefined variable default handling system-wide.

```
$ZUTIL(69,0,n)
$ZU(69,0,n)
```

Parameters

n	A numeric code specifying how to handle undefined variables system-wide. Permitted values are 0, 1, and 2.

Description

$ZUTIL(69,0) is used to specify a system-wide default behavior for handling undefined variables. Setting *n* changes this behavior for local variables, process-private globals, and global variables system-wide; it has no effect on special variables.

$ZUTIL(69,0) only affects variables invoked by subsequent processes; it does not affect the current process. To set handling of undefined variables for the current process, use the $ZUTIL(18) Undefined Variable Behavior function.

The value you assign to the variable *n* becomes the system-wide default behavior for all subsequently initiated processes. The native default setting is 0, which specifies that all undefined variables result in the <UNDEFINED> error message.

Invoking **$ZUTIL(69,0)** without specifying *n* returns the current switch setting.

The system-wide default behavior is configurable. Go to the System Management Portal, select **[Home] > [Configuration] > [Compatibility Settings]**. View and edit the current setting of **Undefined**. The default is 0.

Setting **$ZUTIL(69,0)** overrides the System Configuration default; it does not change the default setting.

MVBasic Undefined Variables

Caché MultiValue Basic (MVBasic) routines are conditionally independent in their handling of undefined variables:

- By default, MVBasic undefined variables are controlled by the System Management Portal **Undefined** configuration setting, the **$ZUTIL(69,0)** system-wide setting and the **$ZUTIL(18)** setting for the current process. They are handled exactly the same as regular Caché variables.
- If **$ZUTIL(68,72,1)** is set for the current process, or **$ZUTIL(69,72,1)** is set system wide, MVBasic undefined variable behavior is independent of the System Management Portal **Undefined** configuration setting, and the **$ZUTIL(69,0)** and **$ZUTIL(18)** settings.

For further details, refer to the Variables page of the *Caché MVBasic Reference*.

Parameters

n

A numeric code that specifies how Caché treats an undefined variable:

0	Issues the <UNDEFINED> error for any undefined variable.
1	Returns a null string for a reference to an undefined variable with subscripts, and issues the <UNDEFINED> error for undefined variables without subscripts.
2	Returns a null string (instead of issuing an <UNDEFINED> error) for any undefined variable.

Integers larger than 2 or smaller than –1 are ignored as no-ops. A value of –1 sets **$ZUTIL(69,0)** to 2.

Notes

The **$DATA** function tests if a specified variable is defined. It returns 0 if the variable is undefined.

The **$GET** function returns a default value if a specified variable is undefined. The basic form of **$GET** returns a null string if the specified variable is undefined.

The **$ZUTIL(69,0)** function defines handling behavior for *all* undefined variables. Setting **$ZUTIL(69,0)** has no effect on **$DATA** or **$GET**.

You can use the **ZWRITE** command to display *defined* variables that have a null string value; **ZWRITE** does not display undefined variables that return a null string.

Example

The following example shows the system-wide and per-process settings for handling undefined variables.

```
NEW fred
SET r69=$ZUTIL(69,0),r18=$ZUTIL(18)  // save local defaults
WRITE !,"system-wide:",$ZUTIL(69,0)," process:",$ZUTIL(18)
SET y=$ZUTIL(18,2)
WRITE !,"system-wide:",$ZUTIL(69,0)," process:",$ZUTIL(18)
SET x=$ZUTIL(69,0,1)
WRITE !,"system-wide:",$ZUTIL(69,0)," process:",$ZUTIL(18)
WRITE !,$GET(fred,"Variable is undefined")
IF fred="" {
   WRITE !,"null string for undefined variable" }
ELSE { WRITE !,"something is wrong here" }
DO $ZUTIL(69,0,r69)  // restore local defaults
DO $ZUTIL(18,r18)
QUIT
```

See Also

- $ZUTIL(18) Undefined Variable Behavior function

- $ZUTIL(68,72) Set MVBasic Undefined Variable Handling function
- $ZUTIL(69,72) Set MVBasic Undefined Variable Handling System-wide function
- ZWRITE command
- $DATA function
- $GET function

$ZUTIL(69,1)

Sets null subscript mode default system-wide.

```
$ZUTIL(69,1,n)
$ZU(69,1,n)
```

Parameters

n	The boolean value that specifies whether or not to enable the setting and referencing of null subscripted globals.

Description

Use **$ZUTIL(69,1)** to specify the system-wide default setting for the null subscript mode switch.

If null subscripted globals are not enabled (*n*=0), attempting to set a null subscripted variable, such as `SET ^x("")=99`, or to reference a null subscript, such as `WRITE ^x("")`, results in a <SUBSCRIPT> error. For details on <SUBSCRIPT> error format, refer to the **$ZERROR** special variable.

If null subscripted globals are enabled (*n*=1), you can set a null subscripted variable, such as `SET ^x("")=99`, just like any other variable. Referencing an undefined null subscripted variable results in a <UNDEFINED> error.

Invoking **$ZUTIL(69,1)** without specifying *n* returns the current switch setting.

Parameters

n

The switch that determines the default setting for the null subscript mode system-wide. 0 = Setting/referencing null subscripted globals disabled. 1 = Setting/referencing null subscripted globals enabled. The native default for this switch is OFF (0).

Notes

The system-wide default behavior is that a null subscript reference causes a <SUBSCRIPT> error. This configuration default cannot be changed using the System Management Portal. Setting **$ZUTIL(69,1)** overrides the system configuration default; it does not change the default setting.

You can also enable/disable this mode on a per-job basis. For information on how to do this, see $ZUTIL(68,1) Null Subscript Mode Process Switch.

The setting of the Null Subscript Mode has no effect on the use of a null string as a subscript index in **$ORDER** and **$QUERY**.

Example

The following example tests the null subscript mode. If null subscripts are enabled locally, it creates some.

```
NullSubsTest
   SET a=$ZUTIL(69,1)
   SET b=$ZUTIL(68,1)
   IF b=1 {
      WRITE !,"null subscripts enabled locally"
UseNullSubs
      WRITE !,"system:",a," local:",b,!
      SET ^x("")=99,^x(0)=0,^x(1)=1
      ZWRITE ^x
      QUIT  }
   ELSEIF a=1 {
      WRITE !,"null subscripts enabled system-wide"
      WRITE !,"system:",a," local:",b
      QUIT  }
   ELSE {
      WRITE !,"will enable null subscripts"
      WRITE !,"system:",a," local:",b
      SET ret=$ZUTIL(69,1,1) ; enable system-wide
      WRITE !,"system:",$ZUTIL(69,1)," local:",$ZUTIL(68,1)
      SET y=$ZUTIL(68,1,1) ; enable for current process
      WRITE !,"system:",$ZUTIL(69,1)," local:",$ZUTIL(68,1)
      GOTO UseNullSubs
    }
    QUIT
```

See Also

- $ZUTIL(68,1) Null Subscript Mode Process Switch function

$ZUTIL(69,2)

Sets sequential file open mode default system-wide.

```
$ZUTIL(69,2,n)
$ZU(69,2,n)
```

Parameters

n	The boolean value that specifies the system-wide default mode for sequential files on **OPEN**. The options are: 0=Read-only, 1=Read/Write.

Description

Use **$ZUTIL(69,2)** to specify the system-wide default open mode for sequential files when an **OPEN** command is issued with no mode parameter specified. The two available modes are open for read-only (0) and open for reading and writing (1). On OpenVMS systems, the default value is 1. On all other platforms, the default value is 0.

Invoking **$ZUTIL(69,2)** without specifying *n* returns the current switch setting.

This configuration default cannot be changed using the System Management Portal.

Setting **$ZUTIL(69,2)** overrides the default system-wide; it does not change the system configuration setting.

This open mode default option can be set for the current process by calling **$ZUTIL(68,2)**.

Parameters

n

The switch that determines the system-wide default mode for sequential files on **OPEN**: 0 means R is the default; 1 means RW is the default. The native default mode for this switch is 0 ("Read").

Example

The following example sets the open mode default, then opens a sequential file:

```
SET x=$ZUTIL(69,2),y=$ZUTIL(68,2)
WRITE !,"system-wide open mode:",x
WRITE !,"current process open mode:",y
IF y=0 {
  SET z=$ZUTIL(68,2,1)
  WRITE !,"Set the process open mode to:",$ZUTIL(68,2)
}
ELSE {
  WRITE !,"Open mode was already set to:",$ZUTIL(68,2)
}
OPEN "c:\myfiles\seqtest1":("NRW"):5
. ...
QUIT
```

See Also

- $ZUTIL(68,2) — Open Mode for Sequential Files function
- OPEN command
- Sequential File I/O in *Caché I/O Device Guide*

$ZUTIL(69,3)

Sets automatic sequential file creation system-wide.

```
$ZUTIL(69,3,n)
$ZU(69,3,n)
```

Parameters

n	A boolean value that specifies whether or not an **OPEN** command should create a new sequential file if the specified sequential file does not exist.

Description

If you attempt to open a sequential file in "W" or "RW" mode but that file does not yet exist, the default behavior (for all platforms except OpenVMS) is for Caché to hang until the file is actually created or the process is resolved by timeout expiration or by calling the RESJOB utility. On OpenVMS systems, the default is to create a new sequential file.

Invoking **$ZUTIL(69,3)** without specifying *n* returns the current switch setting.

This configuration default cannot be changed using the System Management Portal.

You can use **$ZUTIL(69,3)** to set a system-wide automatic file creation mode that causes a sequential file to be created automatically when a user issues the **OPEN** command in "W" or "RW" mode for a file that does not already exist. Setting **$ZUTIL(69,3)** overrides this default system-wide; it does not change the System Configuration setting.

You can set this automatic file creation mode for the current process using **$ZUTIL(68,3)**.

The system-wide or current-process default behavior can be overridden for individual **OPEN** commands using the “E” (or /CREATE) mode parameter. The **OPEN** mode parameters are listed in Sequential File I/O in the *Caché I/O Device Guide*.

Parameters

n

The boolean switch that determines the system-wide default mode for nonexistent sequential files on **OPEN**: 0 = Automatic new file creation disabled. 1 = Automatic new file creation enabled.

See Also

- $ZUTIL(68,3) Automatic Sequential File Creation Mode Process Default function
- OPEN command
- Sequential File I/O in *Caché I/O Device Guide*

$ZUTIL(69,5)

Enables argumentless BREAK processing system-wide.

```
$ZUTIL(69,5,n)
$ZU(69,5,n)
```

Parameters

n	The switch that determines the system-wide default behavior of argumentless **BREAK**: 0 = Caché treats an argumentless **BREAK** as a no-op. 1 = Caché executes **BREAK** on argumentless **BREAK** (the default).

Description

Use **$ZUTIL(69,5)** to enable or disable the processing of argumentless **BREAK** commands system-wide. You can override this system-wide default on a per-process basis by using **$ZUTIL(68,5)**.

Invoking **$ZUTIL(69,5)** without specifying *n* returns the current switch setting.

Parameters

n

The default setting for this switch is 1. InterSystems provides this capability for backwards compatibility.

This configuration default cannot be changed using the System Management Portal.

Setting **$ZUTIL(69,5)** overrides the System Configuration default; it does not change the default setting.

See Also

- BREAK command
- $ZUTIL(68,5) Argumentless BREAK Process Switch function
- Debugging chapter in *Using Caché ObjectScript*

$ZUTIL(69,6)

Sets reliable SET networking mode system-wide.

```
$ZUTIL(69,6,n)
$ZU(69,6,n)
```

Parameters

n	The boolean value that specifies whether or not Reliable SET Networking mode is enabled system-wide.

Description

Use **$ZUTIL(69,6)** to specify the system-wide setting for the Reliable SET Networking mode switch.

When Reliable SET mode is enabled, each request of a **SET**, **KILL**, or **LOCK** command from a DCP client that results in a network error message (such as <PROTECT>, <DIRECTORY>, or <FILEFULL>) waits for a reply before proceeding. When Reliable SET mode is disabled, the local system ignores errors incurred by **SET**, **KILL**, and **LOCK** requests.

By default, Reliable SET mode is disabled, causing the local system to ignore such errors. Setting **$ZUTIL(69,6)** overrides the default system-wide; it does not change the system configuration setting. This configuration default cannot be changed using the System Management Portal.

Invoking **$ZUTIL(69,6)** without specifying *n* returns the current switch setting.

You can also enable or disable this mode on a per-job basis. For information on how to do this, see $ZUTIL(68,6) Reliable SET Networking Mode Process Switch.

Parameters

n

The switch to turn the Reliable SET mode switch on or off. 0 = Disable Reliable SET mode. 1 = Enable Reliable SET mode. The native default for this switch is “false” (0).

See Also

- KILL command
- LOCK command
- SET command
- $ZUTIL(68,6) Reliable SET Networking Mode Process Switch function

$ZUTIL(69,7)

Retains or strips extended global reference from globals system-wide.

```
$ZUTIL(69,7,n)
$ZU(69,7,n)
```

Parameter

n	The boolean switch to set the default for handling extended global references system-wide.

Description

Use **$ZUTIL(69,7)** to specify the system-wide default setting for the Extended Global Reference mode switch. This switch controls whether namespace names are returned as part of a global name by the **$NAME** and **$QUERY** functions. Caché provides for stripping of extended global references for purposes of backwards compatibility. The native default for this switch is 0.

0 = Do not strip the extended global reference; this enables extended reference as a system-wide default.

1 = Strip the extended global reference and return just the global; this disables extended reference as a system-wide default.

Invoking the **ZWRITE** command returns the extended global reference regardless of the **$ZUTIL(69,7)** setting.

Invoking **$ZUTIL(69,7)** without specifying *n* returns the current switch setting.

For further information on extended global references, see Extended Global References in *Using Caché Globals*.

Notes

The system-wide default behavior is configurable. Go to the System Management Portal, select **[Home] > [Configuration] > [Compatibility Settings]**. View and edit the current setting of **RefInKind**. The default is "false".

Setting **$ZUTIL(69,7)** overrides the System Configuration default; it does not change the default setting.

You can also disable or enable this mode on a per-job basis. For information on how to do this, see $ZUTIL(68,7) Strip Extended Global References.

See Also

- $NAME function
- $QUERY function
- $ZUTIL(68,7) Strip Extended Global References function

$ZUTIL(69,8)

Sets ZA and ZD locking modes system-wide.

```
$ZUTIL(69,8,n)
$ZU(69,8,n)
```

Parameter

n	The boolean value to set the system-wide default locking behavior when **ZA** and **ZD** are called.

Description

Invoking **$ZUTIL(69,8,*n*)** causes all **ZA** (ZALLOCATE) and **ZD** (ZDEALLOCATE) resource locking commands in Caché mode routines to act in either Open M [DSM] mode or Caché mode. The **ZA** and **ZD** commands are obsolete; this function is provided for compatibility only.

Invoking **$ZUTIL(69,8)** without specifying *n* returns the current switch setting.

Parameters

n

The boolean switch to set the system-wide default setting for **ZA** and **ZD** behavior.

0 = **ZA** and **ZD** commands act in Caché mode. In Caché mode, **ZA** and **ZD** are synonymous with **LOCK+** and **LOCK–**.

1 = **ZA** and **ZD** commands act in DSM for OpenVMS mode. In DSM-11 mode, **ZA** can only be undone by **ZD** and **LOCK+** can only be undone by **LOCK–**. All **ZA** commands in the same location can be undone with **ZD**.

The default is zero (0). For information about these modes, see the discussion of DSM compatibility modes in Open M Language Compatibility in *Using Caché ObjectScript*.

Notes

The default is 0 (Caché behavior). This configuration default cannot be changed using the System Management Portal. Setting **$ZUTIL(69,8)** overrides the System Configuration default; it does not change the default setting.

ZA and **ZD** modes are provided for compatibility purposes only. Use the **LOCK** +*variable* and **LOCK** –*variable* commands, rather than the obsolete **ZA** and **ZD** commands.

See Also

- LOCK command
- ^$LOCK structured system variable
- ZALLOCATE and ZDEALLOCATE in Open M Language Compatibility in *Using Caché ObjectScript*

$ZUTIL(69,10)

Sets system behavior when journal is full.

```
$ZUTIL(69,10,n)
$ZU(69,10,n)
```

Parameters

n	The boolean value that specifies which system behavior applies when a journal is full.

Description

$ZUTIL(69,10) determines Caché behavior when an error occurs in writing to the journal.

- If this option is set to “true” (1), as soon as the error occurs all global activities that are normally journaled are blocked, which causes other jobs to block. The typical outcome is that Caché goes into a hang state until the journaling problem is resolved, and then resumes running. While Caché is hanging, the administrator can take corrective measures, such as freeing up space on a disk that is full, switching the journal to a new disk, etc. This option has the advantage that once the problem is fixed and Caché resumes running, no journal information has been lost. It has the disadvantage that the system is less available while the problem is being solved.
- If this option is set to “false” (0), when a journaling error occurs journaling is entirely disabled, while Caché continues running as normal. Caché sends a console message to alert the administrator, who can fix the problem and then run ^JRNSWTCH at the console to restart journaling.

The default is zero (0). The native default for this switch is "false" (0); that is, to suspend journaling and allow Caché processing to continue.

Invoking **$ZUTIL(69,10)** without specifying *n* returns the current switch setting.

This system-wide default can be configured. Go to the System Management Portal, select **[Home] > [Configuration] > [Journal Settings]**. View and edit the current setting of **Freeze on error**. The default is "No".

Setting **$ZUTIL(69,10)** changes the system configuration setting shown in the System Management Portal; this change persists across a Caché shutdown and restart.

You can use **$ZUTIL(78,22)** to return journaling information.

Parameters

n

The switch that determines behavior: 0 = Journaling suspends and processing continues. 1 = *System freezes*.

See Also

- $ZUTIL(78,22) Return Journaling Information function
- Journaling in the *Caché Data Integrity Guide*

$ZUTIL(69,11)

Sets Read Line Recall mode system-wide.

```
$ZUTIL(69,11,n)
$ZU(69,11,n)
```

Parameter

n	The boolean value to enable or disable Read Line Recall mode system-wide.

Description

$ZUTIL(69,11) controls Read Line Recall mode system-wide. Read line recall is only used by terminal devices. The default is read line recall enabled.

Read line recall mode is established according to the following rules:

- **$ZUTIL(69,11)** can be used to configure the system-wide default behavior.
- $ZUTIL(68,11) can be used to override this system default for the local process.
- The **OPEN** command sets the read line recall mode for a terminal. You can specify the R protocol (enable) or the N protocol (disable). If neither protocol is specified, **OPEN** takes its setting from the current default established by **$ZUTIL(68,11)** or **$ZUTIL(69,11)**.
- The **USE** command can specify the R protocol (enable) or the N protocol (disable) to change the **OPEN** mode. If neither protocol is specified, **USE** takes its setting from the initial **OPEN** mode value (*not* from the current **$ZUTIL(68,11)** or **$ZUTIL(69,11)** default).
- An implicit open of an active device, such as issuing a **BREAK** command, reopens the device in the same mode as the initial **OPEN** command.

You cannot use **$ZUTIL(69,11)** to override an **OPEN** or **USE** setting for an active terminal. To change read line recall for an already open terminal device, you must explicitly reopen the device. You can use **$ZUTIL(69,11)** or **$ZUTIL(68,11)** to change the default, then issue an **OPEN 0** command, which reopens the active terminal device, applying the current default. See Terminal I/O in *Caché I/O Device Guide* for details on using protocols.

Invoking **$ZUTIL(69,11)** without specifying *n* returns the current switch setting.

Read Line Recall

Read line recall mode provides line recall of editable lines as input for **READ** operations from a terminal. These recallable lines include both previous **READ** input lines and previous command lines. Echoing of input lines is a necessary precondition for read line recall.

Caché supports read line recall for both variable-length terminal reads (`READ var`) and fixed-length terminal reads (`READ var#n`). Caché does not support read line recall for single-character terminal reads (`READ *var`).

For a fixed-length terminal read, the recalled line is truncated to one character less than the number of characters specified in the **READ**. This final **READ** character position is reserved for typing a line termination character, specifying an edit character, or adding one more data character.

When read line recall is active, you can provide input to a **READ** by using the **Up Arrow** and **Down Arrow** keys to recall a previous terminal input line. You can then use the **Left Arrow**, **Right Arrow**, **Home**, and **End** keys to position the cursor for editing the recalled line. You can use the **Backspace** key to delete a character, **Ctrl-X** to delete the entire line, or **Ctrl-U** to delete all of the line to the left of the cursor.

When read line recall is not active, the four **Arrow** keys, the **Home** key, and the **End** key all issue a line termination character. You can use the **Backspace** key to delete a single input character, and **Ctrl-X** (or **Ctrl-U**) to delete the entire input line. Read line recall can be deactivated by using the -R protocol, or by specifying the N, I, S, or T protocols, as described in the Terminal I/O chapter of the *Caché I/O Device Guide*.

Parameters

n

The switch that controls Read Line Recall mode. 0 = Disables Read Line Recall mode as a system default. 1 = Enables Read Line Recall mode as a system default.

The native default for this switch is on (1); that is, enable Read Line Recall.

Notes

This configuration default cannot be changed using the System Management Portal.

Setting **$ZUTIL(69,11)** overrides the System Configuration default; it does not change the default setting.

You can also disable or enable this mode on a per-job and per-device basis. For information on setting this for a job, see $ZUTIL(68,11) Read Line Recall Mode Process Switch.

For information on using the R or N protocol to enable or disable read line recall for a terminal device, see Terminal I/O in *Caché I/O Device Guide*.

See Also

- $ZUTIL(68,11) Read Line Recall Mode Process Switch function
- OPEN command
- READ command
- Terminal I/O in *Caché I/O Device Guide*

$ZUTIL(69,13)

Sets logging of asynchronous SET/KILL errors.

```
$ZUTIL(69,13,n)
$ZU(69,13,n)
```

Parameters

n	The boolean value that specifies whether to log asynchronous **SET**/**KILL** errors in MNETERR.LOG.

Description

$ZUTIL(69,13) controls whether Caché sends network transmission errors that occur during a remote, asynchronous **SET** or **KILL** request to the file MNETERR.LOG in the system manager's namespace.

The corresponding **$ZUTIL(69,14)** function controls whether Caché sends these errors to the operator console.

Invoking **$ZUTIL(69,13)** without specifying *n* returns the current switch setting.

Setting **$ZUTIL(69,13)** overrides the default system-wide; it does not change the system configuration setting. This configuration default cannot be changed using the System Management Portal.

Parameters

n

The switch that controls where Caché sends transmission errors. 0 = Disables writing transmission errors to MNETERR.LOG. 1 = Enables writing transmission errors to MNETERR.LOG. This setting applies only to DCP networks.

The native default for this switch is 0 (OFF). You cannot control this switch on a process level; that is, there is no equivalent **$ZUTIL(68)** call.

See Also

- ZSYNC command
- $ZUTIL(69,14) Send Asynchronous Errors to Operator Console function
- $ZUTIL(68,34) Process Interruptible by Asynchronous Errors function
- $ZUTIL(69,34) Processes Interruptible by Asynchronous Errors function

$ZUTIL(69,14)

Sets sending of asynchronous SET/KILL errors to operator console.

```
$ZUTIL(69,14,n)
$ZU(69,14,n)
```

Parameters

n	The boolean value that specifies whether to send asynchronous **SET**/**KILL** errors to the operator console.

Description

$ZUTIL(69,14) controls whether Caché sends network transmission errors that occur during a remote asynchronous **SET** or **KILL** request to the operator's console.

The corresponding **$ZUTIL(69,13)** function controls whether Caché logs these errors in MNETERR.LOG.

Invoking **$ZUTIL(69,14)** without specifying *n* returns the current switch setting.

Setting **$ZUTIL(69,14)** overrides the default system-wide; it does not change the system configuration setting. This configuration default cannot be changed using the System Management Portal.

Parameters

n

The switch that controls where Caché sends transmission errors: 0 = Disable writing transmission errors to the console. 1 = Enables writing transmission errors to the console. When 1, Caché sends asynchronous network error notification to any operator logged in. When 0, no notification is sent. This setting applies only to DCP networks.

The native default for this switch is off (0); that is, to disable writing of asynchronous **SET**/**KILL** errors to the operator console. You cannot control this switch on a process level 1; that is, there is no equivalent **$ZUTIL(68)** call.

See Also

- ZSYNC command
- $ZUTIL(69,13) Log Asynchronous Errors to MNETERR.LOG function
- $ZUTIL(68,34) Process Interruptible by Asynchronous Errors function
- $ZUTIL(69,34) Processes Interruptible by Asynchronous Errors function

$ZUTIL(69,15)

Sets I/O device disconnect detection system-wide.

```
$ZUTIL(69,15,n)
$ZU(69,15,n)
```

Parameters

n	The boolean value that specifies whether Caché detects I/O device disconnection.

Description

$ZUTIL(69,15) controls whether Caché detects device disconnection for certain types of TCP and terminal devices. This setting specifies how Caché responds to a disconnect of the principal I/O device. Error on disconnect behavior depends on the settings of both **$ZUTIL(69,15)** and **$ZUTIL(69,60)**:

- If **$ZUTIL(69,15,0)** and **$ZUTIL(69,60,0)** the process exits without reporting an error to the application when a disconnect is detected. The default for both functions is 0.
- If **$ZUTIL(69,15,1)** and **$ZUTIL(69,60,0)** the process receives a <DSCON> error when a disconnect is detected during a Caché ObjectScript **READ** or **WRITE** command to the device.
- If **$ZUTIL(69,15,1)** and **$ZUTIL(69,60,1)** the process receives a <DSCON> error immediately when the terminal is disconnected.

You should be aware that when error on disconnect is enabled, a process continues to execute after its principal device has been disconnected. It is the responsibility of the application to detect the disconnect condition and exit gracefully. Use care when enabling error on disconnect. The application must be prepared to recognize the <DSCON> error and handle it appropriately in error traps.

Error on disconnect is only applicable to TCP devices and to terminal devices where a disconnect can be recognized. Examples are modem controlled terminals and Windows Telnet, Windows LAT, and Windows local cterm (TRM:) connections. Error on disconnect is only applicable to the principal device.

This error on disconnect option can be set for the current process by calling **$ZUTIL(68,15)**.

Parameters

n

The switch that controls whether Caché detects I/O device disconnection. 0 = Disables system-wide disconnect detection. 1 = Enables system-wide disconnect detection.

System-wide, disconnect detection is disabled by default. **$ZUTIL(69,15)** returns the current system-wide setting without changing it. **$ZUTIL(69,15,*n*)** sets this option.

This system-wide default can be configured. Go to the System Management Portal, select **[Home] > [Configuration] > [Compatibility Settings]**. View and edit the current setting of **DisconnectErr**. When "true", Caché issues a <DSCON> error when a disconnect is detected during a Caché ObjectScript **READ** or **WRITE** command to the device (equivalent to **$ZUTIL(69,15,1)** / **$ZUTIL(69,60,0)**). When "false", no error is issued when a disconnect is detected. The default is "false".

Setting **$ZUTIL(69,15)** changes the system configuration setting shown in the System Management Portal; this change persists across a Caché shutdown and restart.

See Also

- $ZUTIL(68,15) I/O Device Disconnect Detection function
- $ZUTIL(69,60) Asynchronous Telnet disconnect error handling system-wide function
- TCP Client/Server Communication in *Caché I/O Device Guide*

$ZUTIL(69,19)

Sets DDP password security system-wide.

```
$ZUTIL(69,19,n)
$ZU(69,19,n)
```

Parameters

n	A boolean value that specifies whether to enable or disable password protection for DDP connections.

Description

You can use **$ZUTIL(69,19)** to set password protection for DDP (Distributed Data Processing) network connections system-wide. Setting **$ZUTIL(69,19)** overrides this default system-wide; it does not change the System Configuration setting.

Invoking **$ZUTIL(69,19)** without specifying *n* returns the current switch setting.

This configuration default cannot be changed using the System Management Portal.

Parameters

n

The boolean switch that determines the system-wide default mode for DDP password security: 0 = DDP connections do not need a password to connect to Caché. 1 = DDP connections must use a password to connect to Caché.

In order for a setting of 1 to work, you must obtain the routine ^DDPSEC from the InterSystems Worldwide Response Center.

See Also

•

$ZUTIL(69,20)

Transfers global nodes with null subscripts with DSM-DDP.

```
$ZUTIL(69,20,n)
$ZU(69,20,n)
```

Parameters

n	The boolean value that specifies whether the network allows null subscripts.

Description

With Open M [DSM]-DDP networking, you can transfer global nodes that include null subscripts. **$ZUTIL(69,20)** controls this ability system-wide.

When enabled (*n*=1), Caché can read and write global data with null subscripts across a network connection. When disabled (*n*=0), all data with null subscripts is effectively invisible, and all references to such data generate <SUBSCRIPT> errors. The default is disabled (0). This setting only applies to DCP networks. On ECP, null subscripts are always allowed over the network.

Invoking **$ZUTIL(69,20)** without specifying *n* returns the current switch setting.

Setting **$ZUTIL(69,20)** overrides the default system-wide; it does not change the system configuration setting. This configuration default cannot be changed using the System Management Portal.

Parameters

n

The switch that controls whether Caché can transfer null-subscripted global nodes through Open M [DSM]-DDP. 0 = Disables system-wide transfer of null-subscript nodes. 1 = Enables system-wide transfer of null-subscripted nodes.

By default, the ability to transfer null-subscripted globals is disabled. You cannot control this switch on a process level.

For further details on subscripted globals, see Global Structure in *Using Caché Globals*.

See Also

- $ZUTIL(68,1) Enable or disable use of null subscripts for the current process. function
- $ZUTIL(69,1) Enable or disable use of null subscripts system-wide function

$ZUTIL(69,21)

Sets synchronous commit mode system-wide.

```
$ZUTIL(69,21,n)
$ZU(69,21,n)
```

Parameters

n	A boolean value that specifies whether or not synchronous transaction commit mode is enabled system-wide.

Description

You can use **$ZUTIL(69,21)** to set the synchronous transaction commit mode system-wide. This enables or disables synchronizing **TCOMMIT** with the corresponding journal write operation. Every TCOMMIT command requests a flush of the journal data involved in that transaction to disk. When **$ZUTIL(69,21)** is enabled (set to 1) a **TCOMMIT** does not complete until the journal data write operation completes. When set to 0, **TCOMMIT** does not wait for the write operation to complete. Setting **$ZUTIL(69,21)** changes the system configuration setting and is persistent across Caché shutdowns and startups.

Invoking **$ZUTIL(69,21)** without specifying *n* returns the current switch setting.

This system configuration setting can also be changed using the System Management Portal. Select **[Home] > [Configuration] > [Compatibility Settings]**. View and edit the current setting of **SynchCommit**. The default is "false".

You can also disable or enable this mode on a per-process basis. See $ZUTIL(68,21).

Parameters

n

The boolean switch that determines the system-wide default mode for synchronous transaction commit: 0 = **TCOMMIT** completion does not wait for journal file write completion. 1 = **TCOMMIT** completion waits for the journal file write operation to complete. The default is 0.

See Also

- TCOMMIT command
- $ZUTIL(68,21) Set synchronous commit mode for the current process function

$ZUTIL(69,22)

Sets $X update mode for escape sequences system-wide.

```
$ZUTIL(69,22,n)
$ZU(69,22,n)
```

Parameters

n	A numeric code that specifies the **$X** update mode. Select the value that corresponds to your system platform.

Description

You can control the way Caché updates **$X** when writing a string containing an escape sequence. Default behaviors for various Caché implementations are as follows:

- UNIX® parses the ANSI standard escape sequence and counts the rest of the non-escape characters in the string against **$X**.
- VAX and Alpha do not count any more characters in the string against **$X** as soon as they encounter an escape character (**$CHAR(27)**).
- Open M [DSM] mode counts all characters in a string, including the escape character, against **$X**.
- Open M [DTM] and Open M [MSM] count all characters except for the escape character against **$X**.

$ZUTIL(69,22,*n*) controls this ability system wide.

Invoking **$ZUTIL(69,22)** without specifying *n* returns the current switch setting.

Parameters

n

A numeric code that controls a nondefault, consistent, system-wide way of updating **$X**. 0 = Use UNIX® default behavior on system. 1 = Use Open M [DSM] default behavior on system. 2 = Use Open M [DTM]/Open M [MSM] default behavior on system. 3 = Use OpenVMS Alpha default behavior on system. The default behavior on UNIX® is for *n* to be zero (0). The default behavior on VAX or Alpha for *n* is three (3).

Notes

To be counted correctly, the entire escape sequence must be a single string. For example, the following is one string and is therefore recognized as an escape sequence:

```
WRITE $CHAR(27)_"[0m"
```

However, the following is two strings and is therefore not recognized as an escape sequence:

```
WRITE $CHAR(27),"[0m"
```

See Also

- $ZUTIL(68,22) $X Update Mode for Escape Sequences function
- $X special variable

$ZUTIL(69,24)

Sets $ZF process deletion behavior for OpenVMS STOP/ID system-wide.

```
$ZUTIL(69,24,n)
$ZU(69,24,n)
```

Parameters

n	The switch that controls whether Caché can delete a process that is executing a user-written **$ZF** function. 0 = Enable deletion of Caché processes with STOP/ID. 1 = Disable deletion of Caché processes with STOP/ID.

Description

Using the OpenVMS STOP/ID command to delete a Caché process can cause your system to fail. For this reason, Caché on OpenVMS systems prevent you from deleting Caché processes with STOP/ID.

For processes that are executing **$ZF** functions, you may need to be able to delete a Caché process from OpenVMS. Some users of OpenVMS use a form of the **VIEW** command to makes such processes deletable.

$ZUTIL(69,24,*n*) gives you the same ability on all OpenVMS systems.

$ZUTIL(69,24,1) is enabled automatically at startup. That is, by default, you cannot delete Caché processes using STOP/ID.

Invoking **$ZUTIL(69,24)** without specifying *n* returns the current switch setting.

Setting **$ZUTIL(69,24)** overrides the default system-wide; it does not change the system configuration setting. This configuration default cannot be changed using the System Management Portal.

Notes

$ZUTIL(69,24,*n*) affects only user-written **$ZF** functions. You cannot use STOP/ID to delete processes executing **$ZF(-1)** regardless of the state of **$ZUTIL(69,24)**.

When you use STOP/ID to delete a process executing **$ZF**, a "ghost process" appears in the %SS listing. This ghost process cannot cause the system to hang.

$ZUTIL(69,26)

Sets namespace display in programmer prompt system-wide.

```
$ZUTIL(69,26,n)
$ZU(69,26,n)
```

Parameters

n	The boolean value that specifies whether to display the namespace name at the terminal prompt.

Description

Caché allows you to display the current namespace name for your process in the Caché Terminal prompt. (This setting also controls the display of the current namespace name in Telnet windows.) You control whether the current namespace is displayed, by default, as part of the prompt for all processes on the system when you establish a configuration with the System Management Portal, or a %ZSTART (or ZSTU) user-defined system configuration routine.

You can also use **$ZUTIL(69,26,*n*)** to change whether the system displays the current namespace as a default for all processes at some point after system startup.

Invoking **$ZUTIL(69,26)** without specifying *n* returns the current switch setting.

You can set various values, including the current namespace, as the terminal prompt for the current process using $ZUTIL(186).

Caché maintains the name of the current namespace in the **$ZNSPACE** special variable. The current namespace can be an explicit namespace or an implied namespace.

Parameters

n

A boolean switch that determines whether Caché by default displays the current namespace at the terminal prompt for all processes on the system. 0 = Do not display the current namespace name. 1 = Display the current namespace name (the default).

Issuing **$ZUTIL(69,26,1)** sets the switch. Issuing **$ZUTIL(69,26,0)** clears the switch.

Notes

This system-wide default is configurable. Go to the System Management Portal, select **[Home] > [Configuration] > [Compatibility Settings]**. View and edit the current setting of **NamespacePrompt**. The default is platform-dependent.

Setting **$ZUTIL(69,26)** changes the system configuration setting shown in the System Management Portal; this change persists across a Caché shutdown and restart.

Any change you make with **$ZUTIL(69,26)** affects only processes that log in after you issue **$ZUTIL(69,26)**. That is, if you issue **$ZUTIL(69,26,0)**, only processes that subsequently log in do not display their current namespace. Processes that logged in before you issued **$ZUTIL(69,26)** still display their current namespace in the programmer-mode prompt.

See Also

- $ZUTIL(68,26) Set Terminal Prompt Display of Current Namespace for the Current Process function
- $ZUTIL(186) Set Terminal Prompt Display Values for the Current Process function
- ZNSPACE command
- $ZNSPACE special variable

$ZUTIL(69,27)

Enables or disables network hardening system-wide.

```
$ZUTIL(69,27,n)
$ZU(69,27,n)
```

Parameters

n	The boolean value that specifies whether to enable network hardening system-wide.

Description

You can use **$ZUTIL(69,27)** to enable or disable system-wide network hardening.

Invoking **$ZUTIL(69,27)** without specifying *n* returns the current switch setting.

Caché provides *network hardening* to allow user jobs to continue processing in the event of network transport errors. Network hardening allows global references to remote servers to complete successfully even if network transport errors occur.

When a network transport error occurs, the network transport code re-establishes connections and retransmits network requests. Where the user or application must take some action to restore communications, the client process receives a specific error.

This configuration default cannot be changed using the System Management Portal.

Note: This function provide network hardening support for DCP (Distributed Cache Protocol) distributed data management. It does not provide support for ECP (Enterprise Cache Protocol) distributed data management.

DCP (Distributed Cache Protocol) with network hardening performs multiple retries of a network request. ECP (Enterprise Cache Protocol) does not support network hardening and does not perform multiple retries. ECP issues a <NETWORK> error when encountering a network transport error; the suggested programming practice is to roll back the current transaction when encountering a <NETWORK> error, then retry the transaction. At Caché version 5.0 and subsequent, the default for distributed data caching is ECP. For further details, see ECP and DCP in the *Caché Distributed Data Management Guide*.

Parameters

n

A switch that determines whether Caché by default enables or disables network hardening for the system. 0 = Disable network hardening. 1 = Enable network hardening. The default is 1.

See Also

- $ZUTIL(68,27) Enable/Disable Network Hardening for the Current Process function

$ZUTIL(69,28)

Controls root-level (unsubscripted) global node kills system-wide.

```
$ZUTIL(69,28,n)
$ZU(69,28,n)
```

Parameters

n	An integer code value that specifies whether Caché allows the processes to kill root-level global nodes system-wide. Supported values are 0, 1, and 2.

Description

Setting **$ZUTIL(69,28)** allows you to control whether processes on your system can kill root-level (unsubscripted) global nodes.

Invoking **$ZUTIL(69,28)** without specifying *n* returns the current switch setting.

Parameters

n

An integer code that specifies whether Caché allows the processes to kill root-level global nodes: 0 = Root-level global node kills are allowed. 1 = Root-level global node kills are not allowed. 2 = Root-level global node kills are not allowed from the command line, but are allowed from routines.

The default for the system is zero (0). This means that killing a root-level node does not generate an error. For example, you can execute the following without an error:

```
KILL ^a
```

If you set **$ZUTIL(69,28)** to disallow root-level global node kills, attempting to kill a root-level global node generates a <PROTECT> error. For details on <PROTECT> error format, refer to the **$ZERROR** special variable.

Notes

This system-wide default behavior is configurable. Go to the System Management Portal, select **[Home] > [Configuration] > [Compatibility Settings]**. View and edit the current setting of **GlobalKillEnabled**. The default is "true".

Setting **$ZUTIL(69,28)** overrides the System Configuration default; it does not change the default setting.

Any change you make with **$ZUTIL(69,28)** affects only processes that log in after you issue **$ZUTIL(69,28)**. That is, if you issue **$ZUTIL(69,28,1)**, only processes that log in after the function call was issued generate <PROTECT> errors when attempts are made to kill root-level nodes. Processes that logged in before you issued **$ZUTIL(69,28)** can still kill root-level nodes.

You can also disable or enable this mode on a per-job and per-device basis. For information on setting this for a job, see $ZUTIL(68,28) Control Root (Unsubscripted) Node Kills.

For further details on subscripted global nodes, see Global Structure in *Using Caché Globals*.

See Also

- $ZUTIL(68,28) Control Root (Unsubscripted) Node Kills function
- KILL command

$ZUTIL(69,30)

Sets error handling behavior system-wide.

```
$ZUTIL(69,30,n)
$ZU(69,30,n)
```

Parameters

n	The boolean value that specifies the system-wide default mode for error handling behavior.

Description

Use **$ZUTIL(69,30)** to select either Caché error handling behavior, or DSM error handling behavior system-wide.

When an error handler is invoked in Caché, that error handler remains on the stack of established error handlers. Therefore, if an error occurs when the error handler is executing, that error handler attempts to invoke itself, receives the same error again and enters an infinite loop, unless that error handler explicitly sets **$ZTRAP** to a new value.

When an error handler is invoked in DSM, the error handler is unwound from the stack. Therefore, if an error occurs while the error handler is executing, that error is handled by the previous error handler on the stack.

Invoking **$ZUTIL(69,30)** without specifying *n* returns the current switch setting.

This error-handling option can be set for the current process by calling **$ZUTIL(68,30)**.

Parameters

n

A boolean switch that determines your system's error-handling behavior: 1 = Use Open M [DSM] error-handling behavior. 0 = Use Caché error-handling behavior. The default is Caché behavior.

This configuration default cannot be changed using the System Management Portal. Setting **$ZUTIL(69,30)** overrides the system configuration default; it does not change the default setting.

See Also

- $ZUTIL(68,30) Sets Error-Handling Behavior for the Current Process function
- ZQUIT command
- $ZTRAP special variable
- Error Handling in *Using Caché ObjectScript*

$ZUTIL(69,31)

Sets network locks handling system-wide following a DCP outage.

```
$ZUTIL(69,31,n)
$ZU(69,31,n)
```

Parameters

n	A boolean value that specifies whether or not to reinstate network locks when the network returns after a DCP outage.

Description

You can use **$ZUTIL(69,31)** to set the system-wide handling of network locks when the network returns after a DCP (Distributed Cache Protocol) outage. Setting **$ZUTIL(69,31)** overrides the default system-wide; it does not change the system configuration setting.

Invoking **$ZUTIL(69,31)** without specifying *n* returns the current switch setting.

The configuration default cannot be changed using the System Management Portal.

Parameters

n

The boolean switch that determines the system-wide default mode for reinstating network locks: 0 = Caché network hardening attempts to reinstate the locks after a DCP outage. 1 = Caché “drops” (does not reinstate) network locks when the network returns after a DCP outage. The default is 0.

See Also

- Distributed Cache Protocol (DCP) in the *Caché Distributed Data Management Guide*.

$ZUTIL(69,32)

Sets date range and invalid date behavior system-wide.

```
$ZUTIL(69,32,n)
$ZU(69,32,n)
```

Parameters

n	The boolean value that specifies whether or not to use Caché-style date behavior system-wide.

Description

$ZUTIL(69,32) performs two functions:

- It sets the range of valid dates for **$ZDATE**. This provides compatibility with either Caché or ISM.
- It sets the behavior of **$ZDATE** when an invalid date is input.

$ZUTIL(69,32) only affects the behavior of **$ZDATE**; the other date and time functions are unaffected.

Invoking **$ZUTIL(69,32)** without specifying *n* returns the current switch setting.

Parameters

n

A switch that sets which date range to use. 0= Enables the standard Caché range of valid dates (default). 1 = Enables ISM-compatible range of valid dates.

Notes

Setting the Range of Valid Dates

$ZUTIL(69,32,0) sets the range of valid dates for Caché. This range is from 0 through 2980013, inclusive, which corresponds to dates from 12/31/1840 through 12/31/9999. This range can be restricted by setting the **$ZDATE** *mindate* and *maxdate* parameters.

$ZUTIL(69,32,1) sets the range of valid dates for ISM compatibility. This range is from 1 through 94232, inclusive, which corresponds to dates from 01/01/1841 through 12/30/2098. This date range is set for any **$ZDATE** function call which has three or fewer parameters. If a **$ZDATE** function call has more than three parameters, the valid date range is taken either from the **$ZDATE** *mindate* and *maxdate* parameters (if specified) or from the date range established for the current locale.

The default setting of **$ZUTIL(69,32)** is 0 for all Caché products, and 1 for all ISM products.

Setting the Behavior for Invalid Dates

An invalid date is either not a positive integer, or is not a value within the range of valid dates, as specified above.

$ZUTIL(69,32,0) causes **$ZDATE** to issue the error message <ILLEGAL VALUE> or <VALUE OUT OF RANGE> if you submit an invalid date. The behavior can be overridden by supplying an *erropt* to the **$ZDATE** call.

$ZUTIL(69,32,1) causes **$ZDATE** to return the null string if you submit an invalid date. This behavior is set for any **$ZDATE** function call, regardless of the number of parameters.

This configuration default cannot be changed using the System Management Portal.

Setting **$ZUTIL(69,32)** overrides the System Configuration default; it does not change the default setting.

Calling $ZUTIL(69,32)

When you create a new process, its **$ZDATE** behavior is initialized from the current System Configuration default and **$ZUTIL(69,32)**. Calling **$ZUTIL(68,32)** overrides this behavior for the current process. Changing the setting of **$ZUTIL(69,32)** affects only processes subsequently created, not existing processes.

See Also

- $ZDATE function
- $ZUTIL(68,32) Set Date Range and Invalid Date Behavior function
- More information on locales in the article System Classes for National Language Support

$ZUTIL(69,34)

Sets interruptability of processes by asynchronous errors system-wide.

```
$ZUTIL(69,34,n)
$ZU(69,34,n)
```

Parameters

n	The boolean value that specifies whether or not processes should be interrupted by receiving asynchronous errors system-wide.

Description

$ZUTIL(69,34) allows you to set a system-wide switch that specifies whether all processes that log on after the switch has been set can be interrupted by asynchronous errors. These asynchronous errors are documented in Transaction Processing in *Using Caché ObjectScript*.

Invoking **$ZUTIL(69,34)** without specifying *n* returns the current switch setting.

Setting **$ZUTIL(69,34)** overrides the default system-wide; it does not change the system configuration setting. This configuration default cannot be changed using the System Management Portal.

Parameters

n

A switch that determines whether all new processes can be interrupted by asynchronous errors. 1 = Enables the reception of asynchronous errors (default); disconnect errors are sent asynchronously to the process. 0 = Disables the reception of asynchronous errors; the process only gets the error when it enters the **READ** command.

For this setting to be effective, modem disconnect detection must also be set. See $ZUTIL(69,15) — Modem Disconnect Detection System Default.

Notes

If multiple asynchronous errors are detected for a particular job, the system will trigger at least one asynchronous error. However, there is no guarantee what error will be triggered.

Even if a process has disabled reception of asynchronous errors, an asynchronous error will be triggered the next time the process issues a **ZSYNC** command.

See Also

- ZSYNC command
- $ZUTIL(68,34) Process Interruptible by Asynchronous Errors function

$ZUTIL(69,35)

Sets silent retry for domainspace connection attempts system-wide.

```
$ZUTIL(69,35,n)
$ZU(69,35,n)
```

Parameters

n	A boolean value that specifies whether or not to perform a silent retry for domainspace connection attempts.

Description

You can use **$ZUTIL(69,35)** to set domainspace connection attempt behavior system-wide. When enabled, a failed domainspace connection attempt is automatically retried without issuing an error (a “silent” retry). When disabled, a failed domainspace connection attempt is automatically retried, but an error message is issued.

The default is to issue the error message. Setting **$ZUTIL(69,35)** overrides this default system-wide; it does not change the system configuration setting.

Invoking **$ZUTIL(69,35)** without specifying *n* returns the current switch setting.

This configuration setting cannot be changed using the System Management Portal.

Parameters

n

The boolean switch that determines the system-wide default mode for domainspace connection attempt retry behavior: 0 = Caché returns an error to the user process but continues the attempt to reconnect to the domainspace. 1 = Caché automatically continues to attempt connection to a domainspace until it is successful, without returning an error to the user process during the reconnection attempt(s).

See Also

-

$ZUTIL(69,37)

Sets physical cursor mode system-wide.

```
$ZUTIL(69,37,n)
$ZU(69,37,n)
```

Parameters

n	A boolean value that specifies whether or not physical cursor positioning mode is enabled system-wide.

Description

You can use **$ZUTIL(69,37)** to set the physical cursor positioning mode system-wide. This setting accommodates character sets that use two physical spaces for a character, by determining how Caché backspaces over characters. When enabled, Caché examines a character to see if it occupies two physical positions on the device, and if so, writes two backspaces and two space characters. By default, Caché writes one backspace and one space character.

Setting **$ZUTIL(69,37)** changes the system configuration setting and is persistent across Caché shutdowns and startups.

Invoking **$ZUTIL(69,37)** without specifying *n* returns the current switch setting.

This setting is enabled by default in Japanese locales as the property PhysicalCursor. It is disabled in all other locales. It cannot be set using the System Management Portal.

For further information on Japanese language support in Caché ObjectScript, refer to:

- $ZUTIL(55) to set language mode.
- $ZZENKAKU to convert hiragana and katakana alphabetic characters.
- $WASCII, $WCHAR, and $WISWIDE functions for handling surrogate pairs (a pair of 16-bit Unicode characters that encode a single ideographic character) used in the Japanese JIS standard.

Parameters

n

The boolean switch that determines the system-wide default mode for physical cursor positioning: 0 = space and backspace characters take one physical position. 1 = Caché examines each space or backspace character to see if it occupies two physical positions on the device; if so, Caché writes two backspaces and two space characters.

See Also

- $X special variable
- $Y special variable

$ZUTIL(69,39)

Sets caching for future processes system-wide.

```
$ZUTIL(69,39,n)
$ZU(69,39,n)
```

Parameters

n	The boolean value that specifies whether or not a process has caching enabled. Note that settings are: 0 = caching enabled. 1 = caching disabled.

Description

Use **$ZUTIL(69,39)** to disable or enable client server caching system-wide for any future process. When caching is enabled, Caché will cache data for quicker retrieval.

Invoking **$ZUTIL(69,39)** without specifying *n* returns the current switch setting.

Parameters

n

The boolean switch that determines whether caching is disabled. 0 = caching enabled. 1 = caching disabled. Note that these boolean values are the *opposite* of those found in System Configuration.

Caching enabled (0) is the default setting for the switch. This configuration default cannot be changed using the System Management Portal.

Setting **$ZUTIL(69,39)** overrides this default system-wide; it does not change the System Configuration setting. Setting **$ZUTIL(68,39)** overrides the system-wide setting for the current process.

See Also

- $ZUTIL(68,39) Disable or Enable Caching function

$ZUTIL(69,40)

Sets end-of-file handling for sequential files system-wide.

```
$ZUTIL(69,40,n)
$ZU(69,40,n)
```

Parameters

n	The boolean value that sets end-of-file handling system-wide. 0=Caché format. 1=end-of-file flagging format.

Description

The **$ZUTIL(69,40)** function specifies end-of-file flagging format system-wide. This function is provided to support certain features of Ensemble, and the porting of MSM routines that use the MSM **$ZC** function to check for end of file on

sequential file reads. This MSM function should not be confused with the Caché **$ZC** function (which is an abbreviation for **$ZCHILD**).

$ZUTIL(69,40,1) eliminates the <ENDOFFILE> error for sequential files system-wide. Instead, when the end of a file is reached, the **READ** command returns a null string, the **$ZPOS** special variable is set to "" (the null string), and the **$ZEOF** special variable is set to -1.

Invoking **$ZUTIL(69,40)** without specifying *n* returns the current switch setting.

Parameters

n

The switch that determines whether end-of-files are flagged: 0 = End-of-files are flagged in standard Caché format. 1 specifies that ends of files are flagged in **$ZPOS** and **$ZEOF**.

$ZUTIL(69,40,0) is the default setting for the switch. This option controls the behavior when Caché encounters an unexpected end-of-file when reading a sequential file. When set to 1, Caché sets the **$ZEOF** special variable to indicate that you have reached the end of the file. When set to 0, Caché issues an <ENDOFFILE> error. The default is 0.

This configuration default cannot be changed using the System Management Portal.

Setting **$ZUTIL(69,40)** overrides this default system-wide; it does not change the System Configuration setting. Setting **$ZUTIL(68,40)** overrides the system-wide setting for the current process.

See Also

- $ZUTIL(68,40) End-of-File Handling for Sequential Files function
- READ command
- $ZEOF special variable
- $ZPOS special variable
- Sequential File I/O in *Caché I/O Device Guide*

$ZUTIL(69,42)

Sets $JOB format default system-wide.

```
$ZUTIL(69,42,n)
$ZU(69,42,n)
```

Parameters

n	The boolean value that specifies whether or not to use the standard **$JOB** return string system-wide.

Description

$ZUTIL(69,42,1) permits the **$JOB** special variable to return a string of the format *processid*:*nodename* for all processes system-wide. This function provides for a unique value for **$JOB** on networked systems. This is useful when using **$JOB** to index globals accessed by more than one networked system.

Invoking **$ZUTIL(69,42)** without specifying *n* returns the current switch setting.

Parameters

n

The switch that determines the **$JOB** returned string format. 0 = Standard **$JOB** format. 1 = Format the returned string as *processid:nodename*.

$ZUTIL(69,42,0) is the default setting for the switch.

This configuration default cannot be changed using the System Management Portal.

Setting **$ZUTIL(69,42)** overrides the System Configuration default; it does not change the default setting.

See Also

- $ZUTIL(68,42) $JOB Format for Process function
- $JOB special variable

$ZUTIL(69,43)

Sets clearing of global vectors system-wide.

```
$ZUTIL(69,43,n)
$ZU(69,43,n)
```

Parameters

n	The boolean value that specifies whether or not **$ZUTIL(5)** clears global vectors for the current process.

Description

$ZUTIL(69,43,1) enables the clearing of global vectors system-wide when **$ZUTIL(5)** is issued with the current namespace. This function provides compatibility with **$ZUTIL(5)** behavior before Caché version 3.1.

When enabled, **$ZUTIL(5)** changes to a different namespace and resets the information in **$ZUTIL(39)**. When not enabled, **$ZUTIL(5)** changes to a new namespace, but does not reset **$ZUTIL(39)**.

Invoking **$ZUTIL(69,43)** without specifying *n* returns the current switch setting.

Parameters

n

The switch that determines whether global vectors are cleared. 0 = Do not clear global vectors. 1 = Clear global vectors (**$ZUTIL(5)** behaves as it did before Caché version 3.1).

$ZUTIL(69,43,0) is the default setting for the switch.

This configuration default cannot be changed using the System Management Portal.

Setting **$ZUTIL(69,43)** overrides the System Configuration default; it does not change the default setting.

See Also

- $ZUTIL(5) Display or Switch Namespace function
- $ZUTIL(68,43) Sets Clearing of Global Vectors for the Current Process function

$ZUTIL(69,44)

Sets use of the Nagle algorithm for Telnet transmissions system-wide.

```
$ZUTIL(69,44,n)
$ZU(69,44,n)
```

Parameters

n	A boolean value that specifies whether or not the Nagle algorithm is enabled.

Description

You can use **$ZUTIL(69,44)** to set the use of the Nagle algorithm system-wide. The Nagle algorithm makes Telnet more efficient. It reduces the number of IP packets sent over the network by consolidating messages that are sent within a small time interval into a single IP packet. When the Nagle algorithm is enabled, the operating system waits some interval before actually committing the data from a send command, in the hopes that the application will call send again with more data that can be consolidated with the first.

Setting **$ZUTIL(69,44)** changes the system configuration setting and is persistent across Caché shutdowns and startups. This configuration setting cannot be changed using the System Management Portal.

Invoking **$ZUTIL(69,44)** without specifying *n* returns the current switch setting.

Parameters

n

The boolean switch that determines the system-wide default mode for use of the Nagle Algorithm: 0 = Nagle algorithm disabled. 1 = Nagle algorithm enabled. The default is 0.

See Also

•

$ZUTIL(69,45)

Truncates numbers during string-to-number conversion system-wide.

```
$ZUTIL(69,45,n)
$ZU(69,45,n)
```

Parameters

n	The boolean value that specifies whether or not to truncate numbers system-wide.

Description

The **$ZUTIL(69,45)** function specifies whether or not Caché truncates very large numbers during string-to-number conversions system-wide. The default behavior is *not* to truncate numbers.

Normally, when Caché encounters a number larger than 9223372036854775807E127 (or smaller than -9223372036854775808E127) it converts the number to an IEEE double-precision number. This conversion only occurs during string-to-number conversions. A <MAXNUMBER> error can still occur during arithmetic operations. This conversion

extends the numeric range of available numbers, but reduces the numeric precision by roughly 3 decimal digits. When Caché encounters a number too large to be represented by an IEEE double-precision number, it either issues a <MAXNUMBER> error or returns INF, depending on the setting of **$ZUTIL(68,70)** or **$ZUTIL(69,70)**. For further details on IEEE double-precision numbers, refer to the $DOUBLE function.

When **$ZUTIL(69,45,1)** is set, this conversion to an IEEE double-precision number does not occur. Instead, the number's scientific notation exponent is truncated to the biggest valid exponent value for a Caché floating point number, and the string-to-number conversion continues.

This function is provided for MSM compatibility. Specifying **$ZUTIL(55,8)** to set MSM language mode automatically sets **$ZUTIL(69,45)**. Invoking **$ZUTIL(69,45)** without specifying *n* returns the current switch setting.

Parameters

n

The boolean value that specifies whether number truncation occurs. 0 = preserves large numbers and performs IEEE double-precision conversion on these numbers when necessary (the default); 1 = truncates large numbers during the string-to-number conversion and suppresses IEEE conversion and <MAXNUMBER> errors.

$ZUTIL(69,45,0) is the default setting for the switch in all language modes except MSM mode. This configuration default cannot be changed using the System Management Portal.

Setting **$ZUTIL(69,45)** overrides the System Configuration default; it does not change the configuration setting.

Examples

The following example demonstrates the conversion of a large number to an IEEE double-precision number when **$ZUTIL(69,45,0)** is set (the default).

```
SET rstore=$ZUTIL(69,45)  // get local default
SET rtn=$ZUTIL(69,45,0)
WRITE !,"E127 no $DOUBLE conversion"
WRITE !,92233720368547758O7E127
WRITE !,$DOUBLE(92233720368547758O7E127)
WRITE !,"E128 automatic $DOUBLE conversion"
WRITE !,92233720368547758O7E128
WRITE !,$DOUBLE(92233720368547758O7E128)
SET x=$ZUTIL(69,45,rstore)  // reset to default
```

The following example demonstrates the scientific notation exponent truncation of a large floating-point number when **$ZUTIL(69,45,1)** is set.

```
SET rstore=$ZUTIL(69,45)  // get local default
SET rtn=$ZUTIL(69,45,1)
WRITE !,"E127 no truncation"
WRITE !,92233720368547758O7E127
WRITE !,"E128 automatic truncation to E127"
WRITE !,92233720368547758O7E128
WRITE !,"E128 explicit $DOUBLE conversion"
WRITE !,$DOUBLE(92233720368547758O7E128)
SET x=$ZUTIL(69,45,rstore)  // reset to default
```

See Also

- $DOUBLE function
- $ZUTIL(55) Language Mode Switch function
- $ZUTIL(68,45) Truncate Numbers During String-to-Number Conversion for the Current Process function
- $ZUTIL(68,70) — Set $DOUBLE INF and NAN behavior function
- $ZUTIL(69,70) — Set $DOUBLE INF and NAN behavior system-wide function

$ZUTIL(69,49)

Sets logging of transaction rollbacks system-wide.

```
$ZUTIL(69,49,n)
$ZU(69,49,n)
```

Parameters

n	A boolean value that specifies whether or not to log transaction rollbacks.

Description

You can use **$ZUTIL(69,49)** to set logging of transaction rollbacks system-wide. When enabled, Caché logs transaction rollbacks to the console log file: cconsole.log in the Caché system management directory (Mgr), or to an alternate *filename*.log named by the second piece of the console parameter.

This default console log file location is configurable. Go to the System Management Portal, select **[Home] > [Configuration] > [Memory Settings]**. View and edit the current setting of **ConsoleFile**. By default this setting is blank, routing console messages to cconsole.log in the MGR directory. If you change this setting, you must restart Caché for this change to take effect.

The default is to not log transaction rollbacks. Setting **$ZUTIL(69,49)** overrides this default system-wide; it does not change the system configuration setting.

Invoking **$ZUTIL(69,49)** without specifying *n* returns the current switch setting.

The configuration default cannot be changed using the System Management Portal.

Parameters

n

The boolean switch that determines the system-wide default mode for transaction rollback logging: 0 = rollback logging disabled. 1 = rollback logging enabled. The default is 0.

See Also

- TROLLBACK command

$ZUTIL(69,51)

Sets namespace default directory assignment behavior system-wide.

```
$ZUTIL(69,51,n)
$ZU(69,51,n)
```

Parameters

n	The boolean value that specifies whether or not changing the namespace also changes the default directory.

Description

$ZUTIL(69,51) allows you to specify whether changing your namespace should also change the default directory. Normally, when you change to a different namespace, Caché changes the directory at the operating system level to the default directory for that namespace. This function allows you to change the namespace without changing the directory.

Invoking **$ZUTIL(69,51)** without specifying *n* returns the current switch setting.

Parameters

n

The switch that determines the default behavior when changing namespaces. 0 = Changing the namespace automatically changes the default directory to the directory for that namespace. 1 = Changing the namespace does not change the default directory. The system default is 0.

Notes

$ZUTIL(69,51) sets this behavior on a system-wide basis. You can set this behavior for the current process by calling **$ZUTIL(68,51)**.

The system-wide default behavior is configurable. Go to the System Management Portal, select **[Home] > [Configuration] > [Compatibility Settings]**. View and edit the current setting of **SwitchOSdir**. The default is "false".

Setting **$ZUTIL(69,51)** changes the system configuration setting shown in the System Management Portal; this change persists across a Caché shutdown and restart.

Call **$ZUTIL(168)** to explicitly set the default directory at the operating system level.

See Also

- $ZUTIL(68,51) Namespace Default Directory Assignment for the Current Process function
- $ZUTIL(168) Set Current Working Directory function

$ZUTIL(69,55)

Selects $X/$Y behavior for TCP devices system-wide.

```
$ZUTIL(69,55,n)
$ZU(69,55,n)
```

Parameters

n	A boolean value that specifies the system-wide default behavior for $X and $Y for TCP devices: 0 = Caché sets $X and $Y to conventional device values (terminal emulation). 1 = Caché sets $X and $Y to specialized TCP values used for TCP data transfer output. The default is 0.

Description

You can use **$ZUTIL(69,55)** to specify the use of the $X and $Y special variables for TCP devices on a system-wide basis. By default, Caché sets $X and $Y to logical cursor values for all devices, with $X containing the character count for the current line and $Y containing the line count. This setting is useful when using TCP for terminal emulation. However, when using TCP strictly for data transfer output, you can set this option so that $X contains the byte count in the output

buffer and $Y contains the transmitted buffer count. Use **$ZUTIL(69,55,1)** to enable this specialized $X/$Y behavior for TCP output devices as needed. Specialized $X/$Y behavior is not used for TCP input.

The default setting, in which $X/$Y are set to logical cursor values for TCP terminal emulation does impose a performance penalty on TCP output. For this reason, you may wish to reset this option when using TCP devices for high-speed data transfer output. For example, Caché Server Pages (CSP) does not require $X/$Y support; CSP performance can be improved by calling **$ZUTIL(69,55,1)** to disable default $X/$Y behavior.

For further details on $X/$Y usage with TCP devices, refer to /TCPNOXY and /XYTABLE in the "OPEN and USE Command Keywords for TCP Devices" section of the TCP Client/Server Communication chapter of the *Caché I/O Device Guide*.

The setting of this option establishes system-wide $X/$Y behavior for TCP devices opened by subsequent processes. It has no effect on devices opened by currently active processes. The setting of this option has no effect on $X/$Y behavior of non-TCP devices. You can override this system-wide default for the current process by using **$ZUTIL(68,55)**.

This configuration default cannot be changed using the System Management Portal.

Invoking **$ZUTIL(69,55)** without specifying *n* returns the current switch setting.

See Also

- $ZUTIL(68,55) Select $X/$Y Behavior for TCP Devices function
- $X special variable
- $Y special variable
- TCP Client/Server Communication in *Caché I/O Device Guide*

$ZUTIL(69,60)

Sets handling of asynchronous Telnet disconnect errors system-wide.

```
$ZUTIL(69,60,n)
$ZU(69,60,n)
```

Parameters

n	A boolean value that specifies whether or not processes receive Telnet disconnect errors asynchronously.

Description

You can use **$ZUTIL(69,60)** to set asynchronous disconnect error handling system-wide. **$ZUTIL(69,60)** is only applicable to Telnet connections on Windows. It has no effect on any other device type or operating system.

For **$ZUTIL(69,60)** to be operational, one of the following must be set to enabled (1):

- The System Management Portal Configuration option **ErrorOnDisconnect** must be set to 1 (True). Go to the System Management Portal, **[Home] > [Configuration] > [Compatibility Settings]**. View and edit the current setting of **DisconnectErr**.
- $ZUTIL(69,15) must set to 1.
- $ZUTIL(68,15) must set to 1 for the current process.

The **$ZUTIL(69,60)** default setting is 0 (disabled). Setting **$ZUTIL(69,60,1)** overrides this default system-wide; it does not change the System Configuration setting.

This configuration setting cannot be changed using the System Management Portal.

Invoking **$ZUTIL(69,60)** without specifying *n* returns the current switch setting.

You can use **$ZUTIL(68,60)** to set this behavior for the current process.

Parameters

n

The boolean switch that determines the system-wide default mode for asynchronous disconnect errors: 0 = the process receives a <DSCON> error at the next **READ** or **WRITE** command. 1 = The process receives an asynchronous <DSCON> error immediately when a disconnect occurs on the device. This error occurs at the next command executed. **HANG** commands will be interrupted.

See Also

- $ZUTIL(68,60) — Set Handling of Asynchronous Telnet Disconnect Errors for the Current Process function
- $ZUTIL(68,15) — Modem Disconnect Detection for the Current Process function
- $ZUTIL(69,15) — Modem Disconnect Detection System-wide function

$ZUTIL(69,63)

Enables or disables lowercase “e” as scientific notation symbol system-wide.

```
$ZUTIL(69,63,n)
$ZU(69,63,n)
```

Parameters

n	The boolean value that specifies whether or not Caché should treat lowercase “e” as a scientific notation base-10 exponent symbol. 1 = evaluate “e” as an exponent symbol. 0 = do not evaluate “e” as an exponent symbol. The default is 1.

Description

The **$ZUTIL(69,63)** function disables or enables the evaluation of the lowercase letter “e” as a scientific notation symbol system-wide. **$ZUTIL(69,63)** has no effect on the evaluation of uppercase “E” as a scientific notation symbol.

The system-wide default is to treat both “E” and “e” as scientific notation symbols. You can use **$ZUTIL(69,63)** to reset this system-wide default. For details on overriding this system default for the current process, see $ZUTIL(68,63).

Invoking **$ZUTIL(69,63)** without specifying *n* returns the current switch setting.

See Also

- $ZUTIL(68,63) Disable or Enable “e” as Scientific Notation Symbol function

$ZUTIL(69,66)

Suppress Telnet NUL at end-of-line system-wide.

```
$ZUTIL(69,66,n)
$ZU(69,66,n)
```

Parameters

n	The boolean value that specifies whether or not Caché should suppress NUL following a CR at end-of-line. 1 = suppress NUL. 0 = do not suppress NUL. The default is 0.

Description

The **$ZUTIL(69,66)** function disables or enables the issuance of a NUL character (ASCII 0) following a CR character (ASCII 13) at end-of-line during Telnet transmission. On output, a Telnet network virtual terminal (NVT) performs the following default end-of-line behavior: either issues a CR (carriage return character) followed by a LF (linefeed character), or issues a CR (carriage return character) followed by a NUL character (if no LF is issued). **$ZUTIL(69,66)** can be used to suppress this NUL character.

The system-wide default is to issue both a CR and a NUL. You can use **$ZUTIL(69,66)** to reset this system-wide default. For details on overriding this system default for an individual process, see $ZUTIL(68,66).

$ZUTIL(69,66) affects output only. On input, the NUL character is dropped by default.

The **$ZUTIL(69,66)** setting at the time a Telnet device is opened determines the behavior of the Telnet device. Changing the **$ZUTIL(69,66)** setting has no effect on currently open Telnet devices.

$ZUTIL(69,66) is specific to Windows platforms where the Telnet protocol is implemented internally by Caché. This function has no effect on UNIX® or OpenVMS platforms.

Invoking **$ZUTIL(69,66)** without specifying *n* returns the current switch setting.

See Also

- $ZUTIL(68,66) Suppress Telnet NUL at End-of-line function

$ZUTIL(69,67)

Suppresses or displays the stack and register usage message box system-wide.

```
$ZUTIL(69,67,n)
$ZU(69,67,n)
```

Parameters

n	The boolean value that specifies whether or not Caché should display or suppress the Windows stack and register usage box for a cache.exe process when it encounters an exception. 1 = enable display of the message box. 0 = suppress display of the message box. The default is 0.

Description

The **$ZUTIL(69,67)** function disables or enables the display of a Windows message box. When enabled, this box displays stack and register usage information when a cache.exe process encounters an exception. Because processing cannot proceed

until the user responds to this message box, **$ZUTIL(69,67)** provides the option to suppress the display of this message box to facilitate unattended system operation, such as scripted shutdown and restart. The default is to suppress the message box.

$ZUTIL(69,67) sets this behavior on a system-wide basis. You can set this behavior for the current process by calling **$ZUTIL(68,67)**. Invoking **$ZUTIL(69,67)** without specifying *n* returns the current switch setting.

This configuration default cannot be changed using the System Management Portal.

See Also

- $ZUTIL(68,67) Suppress Message Box Display for the Current Process function

$ZUTIL(69,68)

Enables or disables the encryption of journal files system-wide.

```
$ZUTIL(69,68,n)
$ZU(69,68,n)
```

Parameters

n	The boolean value that specifies whether or not Caché should establish encryption for future journal files. 1 = encrypt journal files. 0 = do not encrypt journal files. The default is 0.

Description

The **$ZUTIL(69,68)** function disables or enables the encryption of journal files system-wide. **$ZUTIL(69,68)** takes effect when the next journal file is initiated.

Invoking **$ZUTIL(69,68)** without specifying *n* returns the current switch setting.

See Also

- $ZUTIL(69,10) Set Full Journal Behavior function
- $ZUTIL(78,22) Return Journaling Information function
- $ZUTIL(78,23) Delete a Journal File function
- $ZUTIL(78,29) Flush Journal Buffer function
- Using ObjectScript for Transaction Processing in *Using Caché ObjectScript*
- Journaling in the *Caché Data Integrity Guide*

$ZUTIL(69,69)

Enables or disables the use of long strings system-wide.

```
$ZUTIL(69,69,n)
$ZU(69,69,n)
```

Parameters

n	The boolean value that specifies whether or not Caché should allocate stack space to support strings over 32K characters in length. 1 = allocate long string stack space. 0 = do not allocate long string stack space. The default is 0.

Description

The **$ZUTIL(69,69)** function disables or enables the use of extremely long strings (up to 3,641,144 characters) system-wide. It does this by allocating stack space to support strings greater than 32K bytes in length.

The system-wide default is to *not* support the use of extremely long strings. This provides for optimal use of virtual memory on Caché implementations where extremely long strings are not used. If this option is off (the default) attempting to allocate a string of greater than 32K characters results in a <STRINGSTACK> error.

This system-wide default is configurable from the System Management Portal. In the System Management Portal select **[Home] > [Configuration] > [Memory and Startup]**. To enable support for long strings system-wide, select the **Enable Long Strings** check box. Then click the Save button. The default is long strings not enabled.

This option cannot be set on a per-process basis. It also does not affect any existing job. Only jobs started after the setting is changed will use the new value.

Invoking **$ZUTIL(69,69)** without specifying *n* returns the current switch setting.

CAUTION: Enabling long strings does *not* change the default length used to read strings from files. The default read length remains 32,767 characters regardless of whether long strings are enabled or not.

See Also

- READ command
- LENGTH function

$ZUTIL(69,70)

Enables or disables $DOUBLE returning INF and NAN values system-wide.

```
$ZUTIL(69,70,n)
$ZU(69,70,n)
```

Parameters

n	A boolean that specifies whether to generate Caché error messages or return INF, –INF, and NAN values for unresolvable IEEE floating point conversions.

Description

The **$ZUTIL(69,70)** function sets the **$DOUBLE** function return value behavior system-wide. If 0, **$DOUBLE** returns INF (infinity), –INF, and NAN (Not A Number) for unresolvable IEEE floating point conversions. If 1, **$DOUBLE** generates Caché errors for unresolvable IEEE floating point conversions.

$ZUTIL(69,70) controls the issuing of INF, –INF, and NAN when a **$DOUBLE** numeric operation cannot be resolved to a numeric value. It does not control the issuing of INF, –INF, and NAN in all cases. **$DOUBLE** always returns INF, –INF, or NAN when you supply one of these strings as the input value, regardless of the **$ZUTIL(69,70)** setting.

Mathematical operations on **$DOUBLE** numbers that result in an INF, –INF, or NAN are controlled by **$ZUTIL(69,70)**. These include arithmetic operations, exponentiation, and logarithmic and trigonometric functions.

Invoking **$ZUTIL(69,70)** without specifying *n* returns the current switch setting.

This system-wide default behavior can overridden for the current process using the **$ZUTIL(68,70)** function. For further details, see $ZUTIL(68,70) — Set $DOUBLE INF and NAN behavior.

Parameters

n

A boolean switch that specifies the **$DOUBLE** mode setting for the current process.

0	**$DOUBLE** returns INF, –INF, or NAN when given an unresolvable numeric expression.
1	**$DOUBLE** generates Caché <MAXNUMBER>, <ILLEGAL VALUE>, and <DIVIDE> errors when given an unresolvable numeric expression. This is the default.

See Also

- $DOUBLE function
- $ZUTIL(68,70) — Set $DOUBLE INF and NAN behavior function

$ZUTIL(69,71)

Sets IP address format system-wide.

```
$ZUTIL(69,71,n)
$ZU(69,71,n)
```

Parameter

n	A boolean value to set IP address format system-wide. 0=IPv6 format disabled (IPv4 only). 1=IPv6 format enabled (both IPv4 and IPv6 supported).

Description

$ZUTIL(69,71) sets the types of IP address formats supported system-wide. This setting affects all processes started after this function call. IP addresses in IPv6 format can be used in TCP/IP connections. When IPv6 is enabled, Caché accepts an IP address on the **OPEN** command in either IPv6 or IPv4 format, or as a host name, with or without domain qualifiers. Caché first checks for an IPv4 format address, then for an IPv6 format address; it accepts the first successful connection.

To handle IPv6 addresses, you must first enable handling of this format by invoking either **$ZUTIL(69,71,1)**, or **$ZUTIL(68,71,1)** for the current process. The default is IPv4 format only. Further details on IPv4 and IPv6 formats can be found in the "Use of IPv6 Addressing" section of the *InterSystems Product Miscellany* article.

$ZUTIL(69,71) can be used to configure the system-wide default behavior. $ZUTIL(68,71) can be used to override this system default for the local process. This configuration setting cannot be changed using the System Management Portal.

Invoking **$ZUTIL(69,71)** without specifying *n* returns the current switch setting.

See Also

- $ZUTIL(68,71) function
- JOB command
- OPEN command
- TCP Client/Server Communication in *Caché I/O Device Guide*

$ZUTIL(69,72)

Sets MVBasic handling of undefined variables system-wide.

```
$ZUTIL(69,72,n)
$ZU(69,72,n)
```

Parameters

n	The boolean value that specifies whether or not Caché should issue an error when an MVBasic routine references an undefined variable. 1 = substitute the empty string for an undefined variable. 0 = issue an <UNDEFINED> error for an undefined variable. The default is 0.

Description

The **$ZUTIL(69,72)** function defines MVBasic behavior when it encounters a reference to an undefined variable. By default, if an MVBasic routine references an undefined variable, Caché generates an <UNDEFINED> error. You can change this default behavior to have Caché substitute an empty string for an undefined variable, without signalling an error. You can use **$ZUTIL(69,72)** to set this behavior on a system wide basis. You can use $ZUTIL(68,72) to set this behavior for the current process.

Setting this option does not affect any existing job. Only jobs started after the setting is changed will use the new value.

Invoking **$ZUTIL(69,72)** without specifying *n* returns the current switch setting.

See Also

- $ZUTIL(18) function
- $ZUTIL(68,72) function

$ZUTIL(71)

Sets date to a fixed value for the current process.

```
$ZUTIL(71,date)
$ZU(71,date)
```

Parameters

date	A positive integer that represents a date in **$HOROLOG** format.

Description

Sets the date internally stored by Caché in the **$HOROLOG** special variable to a specified fixed value for the current process. Calling **$ZUTIL(71,0)** turns off this function, restoring **$HOROLOG** to the current local date.

Using this function to set a fixed value for the date causes all subsequent calls to **$HOROLOG** from that process to return that fixed value as the date. A date set using **$ZUTIL(71,date)** is a fixed date; it does not increment at midnight.

$ZUTIL(71,date) has no effect on the time portion of **$HOROLOG**, which continues to represent the current time.

To restore **$HOROLOG** to the current date, issue **$ZUTIL(71,0)** or **$ZUTIL(71,"")**. These calls revert **$HOROLOG** to the current date.

$ZUTIL(71,date) returns as its function value the previous fixed date setting. Issue **$ZUTIL(71)** with no *date* parameter to display the date value set by a prior **$ZUTIL(71,date)** call. **$ZUTIL(71)** returns zero if **$HOROLOG** is not set to a fixed value, either because you have not called this function since your Caché process was created, or you have already reset **$HOROLOG** by calling **$ZUTIL(71,0)**. Issuing **$ZUTIL(71)** with no *date* parameter has no effect on **$HOROLOG**.

Parameters

date

An integer that falls between 1 (January 1, 1841) and 2980013 (December 31, 9999) inclusive. Setting *date* to 0 (or a non-numeric string value) turns off this function, resetting **$HOROLOG** to a non-fixed value that is the current date for your locale.

Example

The following example shows the relationship between **$ZUTIL(71)** and **$HOROLOG**. This example uses the **$ZDATETIME** function to show that **$ZUTIL(71)** has no effect on the time portion of **$HOROLOG**, and that **$ZUTIL(71)** does not return a time value. Date format 5 is used here because it always returns a 4–digit year:

```
WRITE $ZDATETIME($HOROLOG,5),!
   /* returns the current date and time,
      in this case: Dec 9, 2008 11:21:29  */
SET x=$ZUTIL(71,61400)
   /* Sets $HOROLOG to a fixed date value.
      Sets x = 0 */
HANG 2
WRITE $ZDATETIME($HOROLOG,5),!
   /* returns the fixed value
      of $HOROLOG set by $ZUTIL(71)
      and the current time:
      Feb 8, 2009 11:21:31  */
HANG 2
WRITE $ZDATETIME($HOROLOG,5),!
   /* returns the fixed date value set by $ZUTIL(71)
      and the incrementing current time:
      Feb 8, 2009 11:21:33  */
WRITE $ZDATETIME($ZUTIL(71,0),5),!
  /* reverts $HOROLOG to its default value,
     the current date, which is a non-fixed value.
     $ZUTIL(71) returns the fixed date value set by
```

```
     the prior $ZUTIL(71): Feb 8, 2009  */
HANG 2
WRITE $ZDATETIME($HOROLOG,5),!
   /* returns the current date and time,
      in this case: Dec 9, 2008 11:21:35 */
```

Notes

$ZUTIL(71,*date*) sets the date only for the process that invokes **$ZUTIL(71,*date*)**. All other Caché processes continue to get **$HOROLOG** values that reflect the actual current date.

$ZUTIL(71,*date*) sets the **$HOROLOG** date for all namespaces and all program stack levels. Issuing a **ZNSPACE** command, a **NEW** command, a **KILL** command, or a **QUIT** command has no effect on the value set for **$HOROLOG**.

$ZUTIL(71,*date*) has no effect on **$NOW**, **$ZTIMESTAMP**, or **$ZUTIL(188)**, which continue to return the current date and time.

See Also

- $HOROLOG special variable
- $NOW function
- $ZUTIL(188) Local Date/Time with Fractional Seconds function
- $ZTIMESTAMP special variable

$ZUTIL(78,21)

Searches journal file for open transactions.

```
$ZUTIL(78,21)
$ZU(78,21)
```

Description

Use **$ZUTIL(78,21)** to search back journal files for open transactions.

You must have the **%Admin_Manage:Use** privilege to execute any of the **$ZUTIL(78)** functions. These functions can also be invoked by % routines located in the manager's directory. For further details, refer to the Privileges and Permissions chapter of the *Caché Security Administration Guide*.

Return Values

$ZUTIL(78,21) returns five fields delimited by commas, as follows:

offset,name,counter,tstart_counter,earliest_counter

offset	Journal offset. An integer.
name	Journal filename. A fully qualified pathname.
counter	Journal file counter (pairing journal filename.) An integer.
tstart_counter	Journal file counter of the earliest **TSTART**. An integer. 0 means no **TSTART** has been journaled.
earliest_counter	Journal offset of the earliest **TSTART** in the file. An integer. 0 means no **TSTART** has been journaled.

The string returned by **$ZUTIL(78,21)** might resemble the following example:

```
132800,c:\InterSystems\Cache\mgr\journal\20061204.001,4,4,132376
```

The record indicated by the last two fields may not be a **TSTART**. However, there will be no open transaction preceding that record.

See Also

- TCOMMIT command
- TSTART command
- Using ObjectScript for Transaction Processing in *Using Caché ObjectScript*
- Journaling in the *Caché Data Integrity Guide*

$ZUTIL(78,22)

Returns journaling information.

```
$ZUTIL(78,22,filename,key)
$ZU(78,22,filename,key)
```

Parameters

filename	*Optional* — The full pathname of a journal file, specified as a quoted string. The default is the current active journal.
key	*Optional* — A numeric code for the type of journal information to return.

Description

Use **$ZUTIL(78,22)** to return journaling information. An argumentless **$ZUTIL(78,22)** returns a string of boolean values indicating the status of the current journal. With arguments, this function returns information concerning a named journal file.

You must have the **%Admin_Manage:Use** privilege to execute any of the **$ZUTIL(78)** functions. These functions can also be invoked by % routines located in the manager's directory. For further details, refer to the Privileges and Permissions chapter of the *Caché Security Administration Guide*.

Note: The preferred way to access journal status information is through the $SYS.Journal.System class methods, as described in the *Cache Class Reference*. Use of **$ZUTIL(78,22)** for new coding is discouraged.

Parameters

filename

The full specification of the journal file on which you want information. Specify filename as a quoted string.

key

An integer code that specifies the type of information you want. For an explanation of the *key* codes, see below. You must specify *filename* to specify *key*.

Notes

Current Journal Status: $ZUTIL(78,22)

If you specify **$ZUTIL(78,22)** with no specified *filename*, **$ZUTIL(78,22)** returns status information about the current journal file. It returns a string such as the following:

```
1^0^0^0^0^1
```

These boolean flags have the following meanings:

```
jrnenabled^jrnpaused^jrnsuspended^jrnfail^jrnfrozen^jrnupdates
```

The *jrnenabled* flag indicates whether journaling is currently enabled.

The *jrnpaused* flag indicates that journal rollover (shifting journaling to a new journal file) is in progress.

The *jrnsuspended* flag indicates that a journal I/O error has caused journaling to be suspended. This is the default journaling configuration setting (**Freeze on error** = "No"). When journaling is suspended, global sets and kills that would normally be journaled continue to be performed without being journaling.

The *jrnfail* flag indicates a journal I/O error. In most cases it means that an error retry is in progress.

The *jrnfrozen* flag indicates that a journal I/O error has caused journaling to be frozen. This should only happen when the journaling configuration setting **Freeze on error** is set to "Yes" (the default is "No"). When journaling is frozen, global sets and kills that would normally be journaled instead hang the system until journaling is unfrozen.

The *jrnupdates* flag indicates whether journaling of database updates is enabled (1), or disabled (0).

Specified Journal: $ZUTIL(78,22,filename)

If you specify **$ZUTIL(78,22,filename)**, Caché opens the specified file and reads the header. You must specify a complete file pathname, such as `"C:\InterSystems\Cache\Mgr\Journal\20061206.001"`. This form of **$ZUTIL(78,22)** returns one of the following values:

0	File does not exist.
-1	File is not a journal file.
1	File exists and is a journal file.

To determine the filename of the current journal file, use **$ZUTIL(78,21)**.

$ZUTIL(78,22,filename,key)

$ZUTIL(78,22,filename,key) returns information about the specified journal file, including the prior and subsequent journal files. You must specify *filename* to specify a *key*.

If the specified file is not a journal file, **$ZUTIL(78,22)** returns the previously listed numeric values for non-journals: 0 for a nonexistent file, or -1 for a non-journal file.

If the specified file is a journal file, **$ZUTIL(78,22)** returns the *key* number, a comma, and the value for that key. If there is no value for specified key (for example, *key*=2 and there is no next journal file), Caché just returns the *key* number.

Key	Returned Value
1	Prior journal file pathname. If there is no prior journal file, returns `1,,-1`, in which 1 is the *key* number, the empty string indicates that there is no prior journal file, and the -1 indicates that the reason there is no prior journal file is that the current journal file is the initial journal file.
2	Next journal file pathname. This is the journal file to be used when the current journal file is full. If no next journal file is defined, this function returns just the *key* number.
3	Maximum size of journal file, specified as an integer number of bytes.

The following example shows the values returned by the *key* values 1 through 3. Note that the *key* number itself is the first thing returned, then a comma, then the value. In this case, there is no next journal file, so *key*=2 returns just the *key* number itself; the comma and null value are not returned.

```
SET jrninfo=$ZUTIL(78,21)
SET jrnname=$PIECE(jrninfo,",",2,2)
WRITE !,"Current journal: ",jrnname
WRITE !,"key 1: prior        ",$ZUTIL(78,22,jrnname,1)
WRITE !,"key 2: next:        ",$ZUTIL(78,22,jrnname,2)
WRITE !,"key 3: size:        ",$ZUTIL(78,22,jrnname,3)
```

```
1,C:\InterSystems\Cache\Mgr\Journal\20070823.001

2

3,1073741824
```

See Also

- $ZUTIL(78,21) Search Journal File for Open Transactions function
- Using ObjectScript for Transaction Processing in *Using Caché ObjectScript*
- Journaling in the *Caché Data Integrity Guide*

$ZUTIL(78,23)

Deletes a journal file.

```
$ZUTIL(78,23,filename)
$ZU(78,23,filename)
```

Parameters

filename	The name of the journal file to delete, specified as a quoted string.

Description

You can use **$ZUTIL(78,23)** to delete a journal file from within Caché. This function does not check that the file to be deleted is a journal file, so it may also delete non-journal files or directories.

You must have the **%Admin_Manage:Use** privilege to execute any of the **$ZUTIL(78)** functions. These functions can also be invoked by % routines located in the manager's directory. For further details, refer to the Privileges and Permissions chapter of the *Caché Security Administration Guide*.

Parameters

filename

The name of the file to delete, specified as a quoted string. **$ZUTIL(78,23)** expands filenames to full canonical form. On OpenVMS systems it automatically specifies the latest version of the file.

Return Values

$ZUTIL(78,23) returns one of the following values:

1	Successful file deletion
other values	Failure. A positive number other than 1 or a negative number indicate failure. The exact return value is platform-dependent. Possible reasons are file not found or file locked by another user.

If the file is locked by another process, you cannot delete it with **$ZUTIL(78,23)**. If the specified file is the active journal, a <FUNCTION> error is generated.

Notes

You can also delete a journal file using the $$DELFILE^JRNUTIL(jrnfile) extrinsic function, as described in the *Caché Data Integrity Guide*. Unlike **$ZUTIL(78,23)**, this extrinsic function confirms that the specified file is a journal file before attempting a delete operation.

See Also

- Journaling in the *Caché Data Integrity Guide*

$ZUTIL(78,28)

Returns journal directory block information.

```
$ZUTIL(78,28)
$ZU(78,28)
```

Description

$ZUTIL(78,28) returns information about the journal directory block.

You must have the **%Admin_Manage:Use** privilege to execute any of the **$ZUTIL(78)** functions. These functions can also be invoked by % routines located in the manager's directory. For further details, refer to the Privileges and Permissions chapter of the *Caché Security Administration Guide*.

$ZUTIL(78,28) returns the information about the journal directory block in the following form:

dirblkver^dirblknum^dirblksize

dirblkver	The directory block version
dirblknum	The directory block number
dirblksize	The directory block size in bytes.

The journal directory block in a journal file stores information about the database directories referenced in journal records. The block is updated the first time a global journal SET or KILL occurs in a newly mounted directory.

Every time the block is updated, its version is incremented by 1. This version number is reset only at startup and is not stored in a journal file. The zero-based block number tells where to locate the directory block in a journal file. For example:

```
OPEN jfile:("rf":8192)
USE jfile:blocknumber
READ x
```

See Also

- Using ObjectScript for Transaction Processing in *Using Caché ObjectScript*
- Journaling in the *Caché Data Integrity Guide*

$ZUTIL(78,29)

Flushes journal buffer.

```
$ZUTIL(78,29)
$ZU(78,29)
```

Description

As part of rolling back a failed transaction, you need to flush to disk the journal buffer associated with that transaction. To do so, you use the **$ZUTIL(78,29)** function.

You must have the **%Admin_Manage:Use** privilege to execute any of the **$ZUTIL(78)** functions. These functions can also be invoked by % routines located in the manager's directory. For further details, refer to the Privileges and Permissions chapter of the *Caché Security Administration Guide*.

$ZUTIL(78,29) returns 1 if the buffer flush is successful, or 0 if the buffer flush is not successful.

See Also

- TROLLBACK command
- Using ObjectScript for Transaction Processing in *Using Caché ObjectScript*
- Journaling in the *Caché Data Integrity Guide*

$ZUTIL(82,12)

Redirects I/O operations.

```
$ZUTIL(82,12,n)
$ZU(82,12,n)
```

Parameter

n	The boolean value that specifies whether or not to perform I/O redirection.

Description

You can use **$ZUTIL(82,12)** to put in effect redirection of I/O commands for the current device. Specifying **$ZUTIL(82,12)** without the *n* parameter returns the current status of this I/O redirection switch.

For information on handling a timeout of a **READ** with I/O redirection, refer to $ZUTIL(96,4).

When you turn on redirection for a particular device, **READ** and **WRITE** commands become calls to labels in the mnemonic space routine. Thus, as the following UNIX® example shows, commands of the form:

```
WRITE /?xxx
or
READ /?xxx
```

are treated as reference to tags %xxx in the mnemonic space routine for the current device. To determine the current device, use the **$IO** special variable.

The redirection entry points and call formats for user-written routines for **WRITE** and **READ** are as follows:

```
WRITE STRING = DO wstr(STRING)
WRITE *CHAR = DO wchr(CHAR)
WRITE ! = DO wnl
WRITE ?POS = DO wtab(POS)
WRITE # = DO wff

READ VAR#LEN:TIMEOUT = SET VAR=$$rstr(LEN,TIMEOUT)
READ *VAR:TIMEOUT = SET VAR=$$rchr(TIMEOUT)
```

Parameters

n

The boolean switch that determines redirection: 0 = Redirection not in effect for current device (default). 1 = Redirection in effect for the current device.

See Also

- READ command
- WRITE command
- $ZUTIL(96,4) — Set $TEST to reflect I/O Redirection function
- $IO special variable

$ZUTIL(86)

Returns configuration file pathname and config name.

```
$ZUTIL(86)
$ZU(86)
```

Description

$ZUTIL(86) returns the full pathname of the configuration file used to start the system, and the current config name. The configuration file supplies both the system configuration parameters and the network configuration parameters during startup. In the following example, the configuration file is Cache.cpf.

```
WRITE $ZUTIL(86)
```

returns `C:\InterSystems\Cache\Cache.cpf*CACHE`

If Caché cannot determine the configuration name, **$ZUTIL(86)** returns UNKNOWN. This is not an error in **$ZUTIL(86)** processing, but reflects a problem with Caché installation. The most common reason for this behavior is the use of symbolic links (which Caché does not currently support) during Caché installation.

$ZUTIL(90,4)

Starts up in a specified namespace (UNIX®/OpenVMS).

```
$ZUTIL(90,4,namespace)
$ZU(90,4,namespace)
```

Parameters

namespace	The name of the namespace, specified as a quoted string.

Description

You can use **$ZUTIL(90,4)** on UNIX® or OpenVMS systems to set the Caché system-wide namespace for all subsequent processes. This function has no effect on Windows systems.

You can use the **$ZUTIL(90,10)** function to determine if a namespace is defined. You can use the **$ZNSPACE** special variable to determine your current namespace.

See Also

- ZNSPACE command
- $ZUTIL(5) Display or Switch Namespace function
- $ZUTIL(90,10) Test Whether a Namespace is Defined function
- $ZNSPACE special variable

$ZUTIL(90,10)

Tests whether a namespace is defined.

```
$ZUTIL(90,10,namespace)
$ZU(90,10,namespace)
```

Parameters

namespace	The name of the namespace, specified as a quoted string.

Description

You can test whether a namespace is defined by issuing this function. It returns 0 if the specified namespace is not defined. It returns 1 if the specified namespace is defined. Namespace names are not case-sensitive; by convention, they are specified and displayed in all uppercase characters.

$ZUTIL(90,10) only recognizes explicit namespace names; it does not recognize implied namespace names.

Example

In the following example, the first **$ZUTIL(90,10)** returns 0 (namespace does not exist) and the second **$ZUTIL(90,10)** returns 1 (namespace does exist):

```
WRITE !,$ZUTIL(90,10,"FRED")  ; a nonexistent namespace
WRITE !,$ZUTIL(90,10,"SAMPLES")
```

See Also

- ZNSPACE command
- $ZUTIL(90,4) Start Up in a Specified Namespace function
- $ZNSPACE special variable

$ZUTIL(94)

Broadcasts a message to a specified process.

```
$ZUTIL(94,pid,message,timeout,noxy)
$ZU(94,pid,message,timeout,noxy)
```

Parameters

pid	A Process ID number.
message	The message to send, specified as a quoted string.
timeout	*Optional* — An integer specifying the number of seconds to wait before timeout. If omitted, no timeout occurs.
noxy	*Optional* — A boolean value specifying whether the **$X** and **$Y** values are to be modified in the terminal process receiving the message. If 0, **$X** and **$Y** are updated. If 1, **$X** and **$Y** are not updated. The default is 0. This parameter does not apply to OpenVMS systems, which do not update **$X** and **$Y** following a broadcast message.

Description

Use **$ZUTIL(94)** to send a message to the principal device of the process you indicate.

$ZUTIL(94) passes the message to the target process. The target process then writes the message to its principal output device. The target process must be running Caché. If you send a message to your own process, you do not see the message on the screen of your principal device.

$ZUTIL(94) does not add any carriage controls to the message it sends. To include any carriage control (carriage returns or line feeds), you must include them in the message, using **$CHAR(10)** and **$CHAR(13)**.

$ZUTIL(94) complements **$ZUTIL(9)**. **$ZUTIL(9)** sends a message to a *specified device* while **$ZUTIL(94)** sends a message to the *principal device of a specified process*. Be sure that you use the right function for the right purpose. If you send a process id to **$ZUTIL(9)** or a terminal device name to **$ZUTIL(94)**, you receive a <FUNCTION> error.

Parameters

pid

The Process ID of the target process. The System Management Portal provides a **Processes** option, which list the pids of all running processes. Select **[Home] > [Processes]**.

message

The message to send, specified as a quoted string.

timeout

A timeout in seconds. If **$ZUTIL(94)** is not able to send the message during the timeout period, it ceases attempts to send the message after *timeout* expires. You must specify *timeout* to specify *noxy*. If you specify a *timeout* of < 0, no timeout occurs. If you specify a *timeout* of 0, Caché sets a timeout of 1 second.

noxy

If 0, **$X** and **$Y** are updated in the receiving process to indicate the location of the end of the broadcast message. If 1, **$X** and **$Y** are not updated. The default is 0. This parameter does not apply to OpenVMS systems, which do not update **$X** and **$Y** following a broadcast message.

See Also

- $ZUTIL(9) Broadcast a Message to a Specified Device function
- Terminal I/O in *Caché I/O Device Guide*

$ZUTIL(96)

Returns or sets Caché information.

```
$ZUTIL(96)
$ZU(96)
```

Description

Use the various forms of $ZUTIL(96) to return or set various items of Caché information. The following sections describe $ZUTIL(96):

- $ZUTIL(96,4) Sets $TEST to Reflect Redirection Operations
- $ZUTIL(96,5) Sets $DEVICE
- $ZUTIL(96,9) Returns the Calling Routine Name
- $ZUTIL(96,10) Returns the Calling Routine Namespace
- $ZUTIL(96,14) Returns the Current Device Type

$ZUTIL(96,3)

Return error number for user-defined command.

```
$ZUTIL(96,3,n)
$ZU(96,3,n)
```

Parameter

n	A user-defined error number to return, specified as an expression that resolves to a positive integer.

Description

$ZUTIL(96,3) returns the error message that corresponds to the value of *n*. The error message occurs at the first stack level above the level that was pushed onto the stack by the invocation of a user-defined command. Refer to %ZLANG for further information on user-defined commands in Caché ObjectScript.

See Also

- %ZLANG
- User Defined Code in *Using Caché ObjectScript*

$ZUTIL(96,4)

Sets $TEST to reflect I/O redirection.

```
$ZUTIL(96,4,n)
$ZU(96,4,n)
```

Parameter

n	The boolean value used to set the **$TEST** special variable upon return from I/O redirection code. 0 = Clears **$TEST** (sets to 0) on return from the redirected **READ**. 1 = Sets **$TEST** (sets to 1) on return from the redirected **READ**.

Description

You can specify how to handle the **$TEST** special variable on return from I/O redirection by issuing this function. By default, the value of **$TEST** is preserved (stored and restored) across a call to extrinsic code, such as I/O redirection code. However, you may wish to return the setting of **$TEST** from I/O redirection code. For example, if application code invokes a **READ** command with I/O redirection, and the **READ** times out, this timeout sets **$TEST** in your application. To preserve this setting of **$TEST** upon returning from I/O redirection code, you must invoke **$ZUTIL(96,4,n)** within the I/O redirection code.

For further information on I/O redirection, refer to $ZUTIL(82,12).

Note: **$ZUTIL(96,4,n)** can only be invoked from extrinsic code. It cannot be called directly from the Caché Terminal. Attempting to do so results in a <FUNCTION> error.

See Also

- READ command
- $TEST special variable

$ZUTIL(96,5)

Sets the $DEVICE special variable.

```
$ZUTIL(96,5,string)
$ZU(96,5,string)
```

Parameter

string	The string value used to set the **$DEVICE** special variable. Specified as a quoted string.

Description

You can use **$ZUTIL(96,5)** to set the value of the **$DEVICE** special variable to *string*. By convention, *string* should describe the outcome of an I/O operation as a 3-piece string, in the form:

```
standard_error,user_error,explanatory_text
```

See Also

- $DEVICE special variable

$ZUTIL(96,9)

Returns the calling routine name.

```
$ZUTIL(96,9)
$ZU(96,9)
```

Description

You can return the name of the calling routine name of the last **DO** or user-defined call made to the current routine by issuing this function.

If no calling routine exists, **$ZUTIL(96,9)** returns the null string.

$ZUTIL(96,10)

Returns the calling routine namespace.

```
$ZUTIL(96,10)
$ZU(96,10)
```

Description

You can return the name of the namespace of the calling routine of the last DO or user-defined call made to the current routine by issuing a call to this function.

If no calling routine exists, $ZUTIL(96,10) returns the null string.

$ZUTIL(96,14)

Returns the current device type.

```
$ZUTIL(96,14)
$ZU(96,14)
```

Description

You can return the current device type by issuing a call to this function. If the current input and output devices are not the same, **$ZUTIL(96,14)** returns the current input device.

The device ID of the current device is found in the **$IO** special variable. The **USE** command establishes a device as the current device.

Return Values

The following table lists the values that **$ZUTIL(96,14)** can return and their meanings:

Value	Device Type
0	Sequential file (see Sequential File I/O in *Caché I/O Device Guide*.)
1	Terminal (see Terminal I/O in *Caché I/O Device Guide*.)
2	Spool device (see The Spool Device in *Caché I/O Device Guide*.)
3	Magnetic tape (see Magnetic Tape I/O in *Caché I/O Device Guide*.)
4	System operator's console
5	Pseudo device (device numbers 20 to 46)
6	Null device
7	Spooled virtual console
8	Interjob communication (devices 22 and 225)
9	TCP bindings
10	SPX NTI device [Caché 3.1 does not support SPX]
11	Net BIOs NTI device
12	Named pipe device

See Also

- USE command
- $IO special variable
- I/O Devices and Commands in *Caché I/O Device Guide*

$ZUTIL(110)

Returns the name of the system that is running.

```
$ZUTIL(110)
$ZU(110)
```

Description

$ZUTIL(110) returns a string that contains the name of your current system. This function is supported on different platforms, as follows:

Windows	Same value as returned by GetComputerNameEx(). This system name may contain lowercase and uppercase letters and may be up to 127 characters in length. (In earlier versions of Caché, the value returned was converted to uppercase and limited to 15 characters.)
OpenVMS	Same value as returned by $GETSYI
UNIX®	Same value as returned by uname()

This function is supported on both 8-bit and Unicode implementations of Caché.

To return the current system name as the terminal prompt, use $ZUTIL(186,1).

Example

The following example returns the name of the system from which it was issued.

```
WRITE $ZUTIL(110)
```

See Also

- $SYSTEM special variable

$ZUTIL(114)

Determines Ethernet address.

```
$ZUTIL(114,flag,name)
$ZU(114,flag,name)
```

Parameters

flag	The switch that specifies the information that **$ZUTIL(114)** is to return. Valid values are 0, 1, and 2.
name	*Optional* — Ethernet device name(s).

Description

You can return a string containing ethernet address information by issuing a call to the following function.

$ZUTIL(114,0) returns the address of the primary ethernet device. This primary ethernet device is the first ethernet device found on the device names list with a valid ethernet address. Any ethernet device can be designated the primary ethernet device.

$ZUTIL(114,0,*name*) returns the address of any attached ethernet device specified by *name*. On OpenVMS systems, this is the physical port address of the ethernet device, not the hardware address.

The ethernet address is returned as a string of 12 characters that represent the 48-bit ethernet address.

The *name* is not case sensitive. The maximum length of a device names list is platform-dependent, but the name of an individual device cannot be more than 15 characters in length. An invalid name value results in a <FUNCTION> error.

$ZUTIL(114,0) returns a null string, rather than an ethernet address if:

- The primary ethernet device is not present in the device names list. **$ZUTIL(114,0,*name*)** returns a null string if the named device is not present in the device names list, or has no corresponding ethernet address.
- On Windows systems, the InterSystems Packet Driver is not installed.
- On IBM AIX® systems, the DLPI (Data Link Provider Interface) packages are not installed.
- The ethernet adapters are protected against access by non-root users, and the process invoking **$ZUTIL(114,0)** is not the root user.

$ZUTIL(114,1) returns the current list of attached ethernet device names, delimited by **$CHAR(1)**. The first name in this list is the primary ethernet device.

$ZUTIL(114,2) returns the current list of ethernet device names, delimited by commas.

$ZUTIL(114,2,*name*) replaces the current ethernet device names list with the list specified in *name*; it then returns the ethernet device names list *prior* to the replacement.

Parameters

flag

The switch that specifies the information that **$ZUTIL(114)** is to return. Valid values are:

Code	Description
0	Returns the address of an attached ethernet device. When *name* is specified, it returns the address of the named device. When *name* is not specified, it returns the primary ethernet device.
1	Returns a list of the ethernet device names actually present on the system. This is a subset of the list returned by *flag*= 2. Listed names are separated by the **$CHAR(1)** character.
2	Returns the current list of ethernet device names, as set by Caché startup or by a subsequent invocation of **$ZUTIL(114,2,*name*)**. Listed names are separated by commas.

name

Ethernet device name(s). Valid values are:

When *flag* = 0	*name* is the name of a specific ethernet device. **$ZUTIL(114,0,*name*)** returns the address of the named device.
When *flag*= 2	*name* is an ethernet device names list, enclosed in quotes with individual device names separated by commas. A name list specified in **$ZUTIL(114,2)** cannot contain control characters.

Examples

The following example returns the address of the first ethernet adaptor found:

```
SET Add=$ZUTIL(114,0)
WRITE Add
```

returns 0000C0A5C222.

The following example returns the address of a named ethernet adaptor:

```
SET Add=$ZUTIL(114,0,"\\device\\NBf_SMCISA1")
WRITE ADD
```

returns 0000C0A6C392.

The following example returns a **$CHAR(1)** delimited list of ethernet adaptor names:

```
SET R=$ZUTIL(114,1)
  FOR I=1:1:$LENGTH(R,$CHAR(1)) {
  WRITE $PIECE(R,$CHAR(1),I),!
  }
```

returns:

```
\\Device\\NwlnkNb
\\Device\\Nbf_SMCISA1
\\Device\\Nbf_NdisWan4
```

The following example first returns a comma-delimited list of ethernet device names and then replaces that list of names with the list of names specified:

```
ZZDUMP $ZUTIL(114,2)
```

returns: en0,en1,et0,et1

```
WRITE !,$ZUTIL(114,2,"es0,es1"),"add one"
WRITE !,$ZUTIL(114,1),"check one"
WRITE !,$ZUTIL(114,2,"es2"),"add two"
WRITE !,$ZUTIL(114,1),"check two"
```

returns: en0,en1,et0,et1

```
ZZDUMP $ZUTIL(114,2)
```

returns: es0,es1

See Also

- ZZDUMP command

$ZUTIL(115,11)

Specifies whether a value can be inserted into an identity column.

```
$ZUTIL(115,11,flag)
$ZU(115,11,flag)
```

Parameters

flag	The switch that specifies whether a value may be inserted to an IDENTITY column. Valid values are 1 (can insert a value) and 0 (cannot insert a value). The default is 0.

Description

$ZUTIL(115,11) specifies whether a value can be specified for an identity column. An identity column is defined using the IDENTITY keyword in the CREATE TABLE statement. A value is specified using an INSERT statement. An identity column value must be a unique integer value. For further details on these SQL statements, refer to the *Caché SQL Reference*.

If *flag*=1, you can use the **INSERT** statement to specify an identity column value. If *flag*=0, attempting to specify an identity column value results in an SQLCODE -111 error.

Specifying a **$ZUTIL(115,11)** *flag* value resets the default for the duration of the current process.

Examples

The following example demonstrates the use of the **$ZUTIL(115,11)** function with embedded SQL. The first program below creates the table `MyEmp` containing a defined IDENTITY field:

```
 DO $SYSTEM.Security.Login("_SYSTEM","SYS")
 ZNSPACE "Samples"
 &sql(CREATE TABLE MyEmp (
EmpNum INT NOT NULL,
MyID   IDENTITY NOT NULL,
Name   CHAR(30) NOT NULL,
CONSTRAINT MYEMPPK PRIMARY KEY (EmpNum))
)
IF SQLCODE=-201 {
  WRITE !,"Table already exists" }
ELSEIF SQLCODE'=0 {
  WRITE !,"CREATE TABLE error is: ",SQLCODE
  QUIT }
ELSE {
   WRITE !,"Successful create" }
```

The second program attempts to insert an IDENTITY field value when **$ZUTIL(115,11)** is set to 0 (the default). This insert operation fails with a SQLCODE -111 error:

```
 DO $SYSTEM.Security.Login("_SYSTEM","SYS")
 ZNSPACE "Samples"
 SET Add=$ZUTIL(115,11,0)
&sql(INSERT INTO MyEmp (EmpNum,MyID,Name) VALUES (0000,9999,'Last Employee') )
WRITE !,"INSERT error code is: ",SQLCODE
```

The third program sets **$ZUTIL(115,11)** to 1, and then attempts to insert an IDENTITY field value. This insert operation succeeds. (If you attempt to run this program more than once, it will fail due to a uniqueness constraint.)

```
 DO $SYSTEM.Security.Login("_SYSTEM","SYS")
 ZNSPACE "Samples"
 SET Add=$ZUTIL(115,11,1)
&sql(INSERT INTO MyEmp (EmpNum,MyID,Name) VALUES (0000,9999,'Last Employee') )
WRITE !,"INSERT error code is: ",SQLCODE
```

The last program below can be used to delete the `MyEmp` table, enabling you to rerun this sequence of program examples:

```
 DO $SYSTEM.Security.Login("_SYSTEM","SYS")
 ZNSPACE "Samples"
 &sql(DROP TABLE MyEmp %DELDATA)
IF SQLCODE=-30 {
  WRITE !,"Table not found (already deleted?)" }
ELSEIF SQLCODE'=0 {
  WRITE !,"DROP TABLE error is: ",SQLCODE
  QUIT }
ELSE {
   WRITE !,"Table deleted" }
```

See Also

- SQL: CREATE TABLE statement
- SQL: INSERT statement
- SQL: UPDATE statement
- SQL: Column
- "Defining Tables" chapter in *Using Caché SQL*

$ZUTIL(128,1)

Returns location of last single step during debugging.

```
$ZUTIL(128,1)
$ZU(128,1)
```

Description

This function is invoked during debugging and returns information about the location of the last single step entered.

Return Values

This function returns the following information about the last single step entered:

<stacklevel><xeclevel><lineref><srcoffset><src>

where:

stacklevel	$STACK level of the breakpoint.
xeclevel	XECUTE level of the breakpoint.
lineref	Line reference of the breakpoint.
srcoffset	0-based offset to the location in the source line where the break has occurred.
src	Source line of the breakpoint.

This function is a useful primitive in the development of trace capabilities beyond what is offered by the "T" action.

You can invoke **$ZUTIL(128,1)** only in the condition expression or the execute code associated with a **ZBREAK** breakpoint, or in routines and functions invoked from these contexts.

You must have the **%Development:Use** privilege to execute the **$ZUTIL(128)** function. For further details, refer to the Privileges and Permissions chapter of the *Caché Security Administration Guide*.

Example

Consider the following sample routine ^TEST:

```
TEST  WRITE !,"THIS IS A TEST"
      QUIT
```

We will establish a single step breakpoint definition that causes the information provided by **$ZUTIL(128,1)** to be written out for each executed line of TEST:

```
ZBREAK $:"N"::"WRITE !,$ZUTIL(128,1)"
BREAK "L+"
DO TEST
```

returns:

```
1 0 TEST^TEST 6 TEST WRITE !,"THIS IS A TEST"
THIS IS A TEST
1 0 TEST+1^TEST 1 QUIT
```

See Also

- BREAK command
- ZBREAK command
- "Tracing Execution" in the Debugging chapter of *Using Caché ObjectScript*

$ZUTIL(130)

Sets or returns the domain ID or index.

```
$ZUTIL(130,flag,0,value)
$ZU(130,flag,0,value)
```

Parameters

flag	Specifies which value to set or return.
0	Always the number zero (0).
value	*Optional* — The value to set.

Description

You can set the domain ID or dsindex, or return their current values by issuing a call to this function.

You must have the **%Admin_Manage:Use** privilege to execute **$ZUTIL(130)** functions. This function can also be invoked by % routines located in the manager's directory. For further details, refer to the Privileges and Permissions chapter of the *Caché Security Administration Guide*.

Parameters

flag

A flag that specifies which **$ZUTIL(130)** value to set or return:

1	domain ID
2	domain index (dsindex)

0

This parameter is always the number zero (0).

value

The value to set. If no value is specified, **$ZUTIL(130)** returns the current value of the flag. A domain ID is a string, and thus must be set as a value in quotes. A domain index is an integer, specified without quotes.

Examples

The following example returns the current domain ID and the current domain index:

```
SET x = $ZUTIL(130,1,0)
SET y = $ZUTIL(130,2,0)
WRITE !,"domain ID is ",x
WRITE !,"domain index is ",y
```

The following example sets the domain ID to 1:

```
DO $ZUTIL(130,1,0,"1")
SET x = $ZUTIL(130,1,0)
WRITE !,"domain ID is ",x
```

The following example sets the domain index to 1:

```
DO $ZUTIL(130,2,0,1)
SET y = $ZUTIL(130,2,0)
WRITE !,"domain index is ",y
```

$ZUTIL(132)

Makes the last device in use the principal I/O device.

```
$ZUTIL(132)
$ZU(132)
```

Description

You can make the current device (last device specified in a USE command) the principal I/O device (referenceable by **USE** 0 or **USE $PRINCIPAL**) by issuing a call to **$ZUTIL(132)**. This function makes the current device the principal I/O device, while leaving the former principal I/O device open, and thus capable of being used explicitly by name.

$ZUTIL(132) takes no arguments. It returns 1 on success, 0 on failure, and generates a <FUNCTION> error if any arguments are specified.

See Also

- USE command

- $PRINCIPAL special variable

$ZUTIL(140)

Returns file, directory, and disk information and performs file operations.

```
$ZUTIL(140,1,name)
$ZUTIL(140,2,name,timeflag)
$ZUTIL(140,3,name,timeflag)
$ZUTIL(140,4,name)
$ZUTIL(140,5,name)
$ZUTIL(140,6,name,newname)
$ZUTIL(140,7,name)
$ZUTIL(140,9,name)
$ZUTIL(140,10,name)
$ZUTIL(140,11,source,destination)
$ZUTIL(140,12,name,mode)
$ZUTIL(140,13,dir)
$ZUTIL(140,14,source,destination)

The $ZU abbreviation can be used with above syntax.
```

Parameters

name	The pathname of the file or directory on which to perform the operation, specified as a quoted string.
timeflag	*Optional* — A boolean value specifying how to return a time value. The available options are local (0), or UTC (1). The default is 0.
newname	For **$ZUTIL(140,6)**, the filename used to rename the file. Specified as a quoted string.
source	For **$ZUTIL(140,11)** and **$ZUTIL(140,14)**, the name of the file or directory used as the source for a copy operation, specified as a quoted string.
destination	For **$ZUTIL(140,11)** and **$ZUTIL(140,14)**, the name of the file or directory used as the destination (target) for a copy operation, specified as a quoted string.
mode	For **$ZUTIL(140,12)**, an integer specifying the operating system permissions to check. Available values are 0 through 7, which correspond to all possible bit mask combinations.
dir	For **$ZUTIL(140,13)**, a disk name or directory pathname, specified as a quoted string.

Description

$ZUTIL(140) provides a (largely) platform-independent interface to retrieve operating system information, such as attributes of a file or directory, and to create or delete files or directories. You can return a variety of file attributes for a specified file or directory using this function. You can test for the existence of a file and check its operating system level access permissions. You can rename, copy, or delete a file. You can create or delete a directory, or copy the files from one directory to another. You can return the time stamp for the creation or most recent modification of a file or directory. You can return the current size of a file. You can return the currently available disk space and block size — information that may be vital when creating or copying a file.

Parameters

flag

A number specifying which file or directory operation to perform. The following table shows the **$ZUTIL(140)** flags recognized by Caché:

Flag	Operation
1	Returns file size in bytes. If an error occurs, **$ZUTIL(140)** returns the operating system's error code as a negative number. Thus, negative values are platform-dependent. The following are common error codes returned by Windows systems: –2 = file not found. Returned if the named element does not exist. –3 = pathname not found. Returned if the pathname ends in a directory with a trailing slash. –5 = access denied. Windows systems return -5 if the named item is a directory, not a file. UNIX® systems return the size of the directory rather than issuing an error code. –12 = invalid access. Returned if user does not have read permission. –123 = invalid formatting. Returned if a valid filename is followed by a trailing slash.
2	Returns modification date/time in $HOROLOG format. Returns -2 if the specified file or directory does not exist. On Windows systems, -2 is also returned if a trailing slash is specified. Using the optional *timeflag*, you can specify whether to return the modification time value in local timezone time or Universal time (UTC). Universal time is (for most purposes) equivalent to Greenwich Mean Time (GMT).
3	Returns creation date/time in $HOROLOG format. Returns -2 if the specified file or directory does not exist. On Windows systems, -2 is also returned if a trailing slash is specified. Using the optional *timeflag*, you can specify whether to return the creation time value in local timezone time or Universal time (UTC). Universal time is (for most purposes) equivalent to Greenwich Mean Time (GMT).
4	Tests whether the named item exists. Returns 0 if the specified pathname exists, for either a file or a directory. Returns -2 if the specified item does not exist. On Windows systems, a directory can be specified with or without a trailing slash. On Windows systems, returns -123 if a valid filename is followed by a trailing slash.
5	Deletes the named file. Can delete a read-only file if user has privileges to modify the directory in which the file resides. Returns 0 when successful. Returns -2 if *name* does not exist. Returns -5 if *name* is a directory.
6	Renames the file specified in *name* with the name specified in *newname*. May move the file to a different location, if the host operating system supports it. Returns -2 if *name* does not exist. Returns -32 if *name* is a directory.
7	Returns file attributes as a bit map. See $ZUTIL(140,7). Returns -2 if the specified file does not exist. On Windows systems, a directory can be specified with or without a trailing slash. On Windows systems, returns -123 if a valid filename is followed by a trailing slash.
8	[unused]

Flag	Operation
9	Creates the directory specified in *name*. Returns 0 when successful, or a negative integer indicating an error. Error return values are supplied by the operating system, and are thus platform dependent. On Windows, returns -183 if the specified directory already exists or could not be created. On UNIX®, returns -17 if the specified directory already exists or could not be created. On OpenVMS, returns a negative integer error code if the specified directory could not be created; returns 0 if the specified directory already exists.
10	Deletes the directory specified in *name*. Returns 0 when successful; returns -2 if the specified directory does not exist.
11	Copies a file or directory from *source* to *destination*. This operation overwrites any prior contents of *destination*. **$ZUTIL(140,14)** is similar, but it does not overwrite *destination*. The *source* must exist. If a *destination* file does not exist, it will be created. If *source* is a file, *destination* may be a file or an existing directory. If *source* is a directory, *destination* must be an existing directory. In this case, all files in the *source* directory are copied to the *destination* directory, but subdirectories in *source* are not copied. Returns 0 when successful; returns a negative integer upon failure.
12	Check a file's access permissions: `$ZUTIL(140,12,`*`name`*`,`*`mode`*`)`. The *mode* bit mask has the following values: 0=file exists, 1=execute permission, 2=write permission, 4=read permission; values 3, 5, 6, and 7 represent combinations of these values. Returns 0 when successful (bit mask matches file's permissions); returns -2 if the file does not exist, -13 if not all specified modes are permitted. **$ZUTIL(140,12)** *does not* change a file's modification date/time.
13	Returns the amount of disk space available on the *dir* disk as four numeric values separated by commas: Free disk space available to the caller. Windows: space available to the caller in bytes (limited if there is a quota for the user). UNIX®: space available to non-privileged users in blocks. OpenVMS: space available to the caller in blocks. Total free disk space. Windows: in bytes. UNIX® or OpenVMS: in blocks. Total disk space. Windows: in bytes. UNIX® or OpenVMS: in blocks. Size of a block. Windows: defaults to 1. UNIX® or OpenVMS: in bytes.
14	Appends the contents of the *source* file to the *destination* file, or copies files from the *source* directory to the *destination* directory. This operation retains any prior contents of *destination*. **$ZUTIL(140,11)** is similar, but it overwrites the prior contents of *destination*. The *source* must exist. If a *destination* file does not exist, it will be created. If *source* is a file, *destination* may be a file or an existing directory. If *source* is a directory, *destination* must be an existing directory. In this case, all files in the *source* directory are copied to the *destination* directory, but subdirectories in *source* are not copied. Returns 0 when successful; returns a negative integer upon failure.

File and Directory Names

The *name*, *newname*, *source* and *destination* parameters all specify a file or directory pathname as a quoted string. Specify these parameter values in the standard syntax of file and directory names for the host operating system.

Windows pathnames can use either the slash (/) or backslash (\) separator character. When specifying a Windows directory, do not append a trailing slash separator.

On Windows and UNIX® systems you can use the following standard pathname symbols: a single dot (.) to specify the current directory, or a double dot (..) to specify its parent directory. For the directory functions on OpenVMS, if *name* does not contain the directory delimiters "[]" or "<>" it is assumed to be a subdirectory of the current directory (i.e. "abc" is equivalent to "[.abc]"). For file deletion on OpenVMS, an explicit file version number must be supplied.

Directory in $ZUTIL(140,13)

$ZUTIL(140,13) uses the *dir* directory pathname to provide the name of the disk. It must, therefore contain the disk name, for example “c:”. *dir* can optionally supply a longer directory pathname, which is validated, but nothing beyond the disk name affects the values returned. If *dir* is a nonexistent disk name or an invalid directory pathname, Caché returns -3. If *dir* is a defined but unmounted disk, Caché returns -21. If *dir* is a file pathname, Caché returns -267. If *dir* contains a wildcard character (*), Caché returns -123.

Examples

The following program example demonstrates the values returned by several of the **$ZUTIL(140)** flag options for a file:

```
SET fname=$ZSEARCH("*cache.lck")
WRITE !,"file path:",fname
WRITE !,"1:",$ZUTIL(140,1,fname)
WRITE !,"2:",$ZUTIL(140,2,fname)
WRITE !,"3:",$ZUTIL(140,3,fname)
WRITE !,"4:",$ZUTIL(140,4,fname)
WRITE !,"7:",$ZUTIL(140,7,fname)
WRITE !,"12:",$ZUTIL(140,12,fname,0)
```

The following program example demonstrates the values returned by **$ZUTIL(140,13)**:

```
SET dir="c:"
SET by=" bytes",bl=" blocks"
SET val=$ZUTIL(140,13,dir)
WRITE !,"string is: ",val
IF $PIECE(val,",",4)=1 {
  WRITE !,"This is a Windows system"
  WRITE !,$PIECE(val,",",1),by," free space available"
  WRITE !,$PIECE(val,",",2),by," total free space"
  WRITE !,$PIECE(val,",",3),by," total disk space"
  }
ELSE { WRITE !,"This is a UNIX or OpenVMS system"
  WRITE !,$PIECE(val,",",1),bl," free space available"
  WRITE !,$PIECE(val,",",2),bl," total free space"
  WRITE !,$PIECE(val,",",3),bl," total disk space"
  WRITE !,$PIECE(val,",",4),by," is the block size"
  }
```

See Also

- $ZSEARCH function
- $ZUTIL(140,7) Return File Attributes function
- $HOROLOG special variable

$ZUTIL(140,7)

Returns a bitmap of file attributes.

```
$ZUTIL(140,7,filename,1)
$ZU(140,7,filename,1)
```

Parameter

filename	The pathname of a file or directory, specified as a quoted string.
1	*Optional (OpenVMS only)* — When the number 1 is specified, returns the VMS record format for the file as a string. This allows you to determine what open mode to use to read the file. Possible return values are UDF, FIX, VAR, VFC, STM, STMLF, and STMCR.

Description

For **$ZUTIL(140)** with second parameter values other than 7, refer to $ZUTIL(140).

Issuing **$ZUTIL(140,7,*filename*)** returns a bitmap that specifies several attributes of the file. The bitmap format is platform-dependent. A value of 1 for a bit specifies that the attribute applies to the file.

$ZUTIL(140,7) returns a negative error code value if an error occurred (the error code is supplied by the host operating system).

Windows bit map:

Bit Position	Meaning
1	Read-only
2	Hidden
4	System
8	
16	Directory
32	Archive
64	Device
128	Normal
256	Temporary
512	Sparse File
1024	Reparse Point
2048	Compressed
4096	Offline
8192	Not Content Indexed
16384	Encrypted

UNIX® mode bit map:

Bit Position	Meaning
1	execute permission for others
2	write permission for others
4	read permission for others
8	execute permission for group
16	write permission for group
32	read permission for group
64	execute permission for owner
128	write permission for owner
256	read permission for owner
512	
1024	set groupid
2048	set userid
4096	

0xF000 is a mask for file type:

1: fifo

2: character special

4: directory

6: block special

8: regular

10: symbolic link

12: socket

OpenVMS bit maps:

For OpenVMS, this is the file protection. It consists of four 4-bit fields, accessed by the following hexadecimal masks:

000F: system privileges

00F0: owner privileges

0F00: group privileges

F000: world privileges

Each of these subfields can contain the following bit values:

1: no read access

2: no write access

4: no execute access

8: no delete access

Examples

The following Windows example shows the values returned for a directory (“samples”) and a database file “CACHE.DAT”:

```
ZNSPACE "%SYS"
SET dir=$ZUTIL(168)  ; determine Caché directory
SET sampdir=dir_"samples"
WRITE !,$ZUTIL(140,7,sampdir)
WRITE !,$ZUTIL(140,7,sampdir_"\CACHE.DAT")
```

The first **$ZUTIL(140,7)** returns 16 (Directory). The second **$ZUTIL(140,7)** returns 8224, indicating that the file is 32 (Archive) and 8192 (not content indexed).

This example requires that UnknownUser have assigned the %DB_CACHESYS role.

See Also

- $ZUTIL(140) Return File Attributes function

$ZUTIL(147)

Handles spaces in pathnames for the host platform.

```
$ZUTIL(147,pathname)
$ZU(147,pathname)
```

Parameters

pathname	A filename or pathname, specified as a quoted string. You cannot supply a Windows *pathname* as a doubly-quoted string.

Description

The **$ZUTIL(147)** function handles spaces in pathnames as appropriate to the host platform. If *pathname* contains a space character, pathname handling is platform-dependent.

- OpenVMS permits space characters in pathnames; **$ZUTIL(147)** performs no special processing and returns the pathname unchanged.
- UNIX® only permits space characters in quoted pathnames; if a pathname containing spaces **$ZUTIL(147)** returns the pathname enclosed in double quotes (“*pathname*”). If a pathname does not contain spaces, **$ZUTIL(147)** returns it unchanged. **$ZUTIL(147)** performs no other pathname validation.
- Windows: **$ZUTIL(147)** strips spaces from the supplied pathname of an existing file. If a pathname does not contain spaces, **$ZUTIL(147)** returns it unchanged in all cases. On Windows systems, **$ZUTIL(147)** validates pathnames that contain spaces. If a pathname containing spaces does not exist, **$ZUTIL(147)** returns the pathname unchanged (with its blank spaces), with the entire pathnames enclosed in double quotes (“*pathname*”). If a pathname containing spaces exists, **$ZUTIL(147)** returns the short form pathname with spaces removed, such as the following:

  ```
  USER>WRITE $ZUTIL(147,"C:\My Test File.txt")
  C:\MYTEST~1.TXT
  ```

  In this case, the filename is truncated to the first six non-blank characters, and a tilde (~) and an ordinal number are appending to distinguish similarly named files. For further details on Windows handling of pathname characters, specify `cmd /?` at the Windows command prompt.

$ZUTIL(147) is commonly used with the **$ZF(-1)** and **$ZF(-2)** functions.

Examples

The following Windows example shows how **$ZUTIL(147)** handles existing and nonexistent pathnames with and without spaces:

```
USER>WRITE $ZUTIL(147,"C:\MyTest.txt") /* Existing no blanks */
C:\MyTest.txt
USER>WRITE $ZUTIL(147,"C:\FakeFile.txt") /* Non-Existing no blanks */
C:\FakeFile.txt
USER>WRITE $ZUTIL(147,"C:\Fake File.txt") /* Non-Existing 1 blank */
"C:\Fake File.txt"
USER>WRITE $ZUTIL(147,"C:\My Test.txt") /* Existing 1 blank */
C:\MYTEST~1.TXT
USER>WRITE $ZUTIL(147,"C:\My Test Doc.txt") /* Existing 2 blanks */
C:\MYTEST~2.TXT
USER>WRITE $ZUTIL(147,"C:\My Test File.txt") /* Existing 2 blanks */
C:\MYTEST~3.TXT
USER>
```

The following example uses **$ZUTIL(147)** to handle a pathname for **$ZF(-1)**. A pathname containing spaces is handled as appropriate for the host platform. A pathname that does not contain spaces is passed through unchanged.

```
SET x=$ZF(-1,$ZUTIL(147,"C:\My Test.txt"))
```

See Also

- $ZF(-1) function
- $ZF(-2) function
- $ZUTIL(12) Translate filename to canonical form function

$ZUTIL(158)

Displays currently installed printers.

```
$ZUTIL(158,0)
$ZUTIL(158,1,n)

$ZU(158,0)
$ZU(158,1,n)
```

Parameters

n	A sequential integer (counting from 1) assigned to a printer.

Description

The **$ZUTIL(158,0)** function returns the number of printers currently installed on your system, counting from 1.

The **$ZUTIL(158,1,n)** function returns the pathname of the printer currently installed on your system that corresponds to *n*. Caché counts printers from 1, and assigns a sequential integer to each. If *n* is a number that does not correspond to a printer, Caché issues a <FUNCTION> error.

Example

The following example returns the total number of installed printers, then returns the pathname for each printer.

```
SET x=$ZUTIL(158,0)
WRITE !,"The number of printers is: ",x
WHILE x>0 {
 WRITE !,"Printer #",x," is ",$ZUTIL(158,1,x)
 SET x=x-1 }
```

See Also

- JOB command
- "Specifying Terminals and Printers by Device Name" in the I/O Devices and Commands chapter of *Caché I/O Device Guide*

$ZUTIL(168)

Returns location of current working directory, or sets current working directory.

```
$ZUTIL(168,dir)
$ZU(168,dir)
```

Parameters

dir	*Optional* — Name of directory to set as working directory for sequential file output. Specified as a quoted string.

Description

$ZUTIL(168) (with no arguments) returns a string that contains the location of your current working directory for sequential file output. (Note that this directory should not be confused with the default directory for globals in the current namespace.)

$ZUTIL(168,dir) sets your current working directory for sequential file output to the *dir* directory. It returns the pathname of the old working directory. If the directory cannot be set as the current working directory, **$ZUTIL(168)** issues a <DIRECTORY> error, and ^SYSLOG will show the cause of the error.

Note: On Windows systems, **$ZUTIL(168,dir)** sets as the current directory a directory name entirely in lowercase letters, regardless of the case of *dir*. Because Windows and Caché ObjectScript are not case-sensitive, this is usually irrelevant. However, this may affect some Windows applications (such as Java) which are case-sensitive.

You can use the wildcard options of the **$ZSEARCH** function to locate an existing directory. You can use **$ZUTIL(140)** to create a new directory, or to return information about an existing file or directory.

Parameters

dir

The name of the directory to set as the working directory. The directory pathname must be specified as a quoted string. On Windows and UNIX® systems you can also use the following standard pathname symbols: a single dot (.) to specify the current directory, or a double dot (..) to specify its parent directory. Remember, when setting a working directory, **$ZUTIL(168,dir)** return the *previous* directory setting.

Examples

```
WRITE $ZUTIL(168)
```

returns something like: c:\InterSystems\Cache\mgr\user\. (The actual directory will differ on different installations of Caché.)

The following example returns the current working directory, then changes the directory to the samples directory.

```
WRITE !,$ZUTIL(168)
ZNSPACE "%SYS"
SET dir=$ZUTIL(168)
SET newdir=$ZUTIL(168,dir_"samples")
WRITE !,$ZUTIL(168)
```

returns something like:

c:\InterSystems\Cache\mgr\user\

c:\InterSystems\Cache\mgr\samples\

This example requires that UnknownUser have assigned the %DB_CACHESYS role.

See Also

- $ZSEARCH function
- $ZUTIL(68,51) Namespace Default Directory Assignment function
- $ZUTIL(69,51) Namespace Default Directory Assignment function
- $ZUTIL(140) Return File Attributes, Create or Delete a Directory function

$ZUTIL(186)

Sets display in programmer prompt for the current process.

```
$ZUTIL(186,n)
$ZU(186,n)
```

Parameters

n	An integer code or a comma-separated list of integer codes that specify what element(s) to include in the terminal prompt. Integer codes should be specified in the order in which you wish the elements to be displayed.

Description

The **$ZUTIL(186)** function specifies what information should appear in the programmer-mode prompt for Caché Terminal.

Issuing **$ZUTIL(186)** (with no option) returns the current state of the prompt setting as a comma-separated list of integer codes. Like all $ZUTIL functions, when you issue **$ZUTIL(186,n)**, it returns its integer code setting before the current operation.

Issuing **$ZUTIL(186,0)** clears the setting, so that no prompt information is displayed.

The initial value of this setting depends on the both the system platform and the system-wide default setting. The system default setting for Windows systems is to display the current namespace at the terminal prompt. You can enable or disable the current namespace display in the terminal prompt for the current process using either **$ZUTIL(186)** or **$ZUTIL(68,26)**.

To control whether the current namespace name appears by default as part of the prompt for *all* processes on the system, use any one of the following:

- Go to the System Management Portal, select **[Home] > [Configuration] > [Compatibility Settings]**. View and edit the current setting of **NamespacePrompt**. The default is platform-dependent.
- $ZUTIL(69,26) — Enable/Disable System Namespace Display Default function.
- The **%ZSTART** user-defined system startup routine (**ZSTU** startup routines are also supported).

Parameters

n

The integer code that specifies what information to display as part of the process' programmer mode prompt.

0	Reset the prompt.
1	Host name, also known as the current system name. The name assigned to your computer. For example, `LABLAPTOP>`. This is the same for all of your terminal processes. Refer to $ZUTIL(110) for further details.
2	Namespace name. For example, `%SYS>`. To set your terminal prompt to just the current namespace name, you can use **$ZUTIL(68,26,1)**. The current namespace name is contained in the **$ZNSPACE** special variable, and can be changed using the **ZNSPACE** command. It can be an explicit namespace name or an implied namespace name.
3	Config name. The name of your Caché installation. For example, `CACHE2>`. This is the same for all of your terminal processes.
4	Current time, expressed as local time in 24-hour format with whole seconds. For example, `15:59:36>`. This is the static time value for when the prompt was returned. This value changes for each prompt.
5	pid. The Process ID for your terminal. For example, `2336>`. This is different for each terminal process. This value can also be returned from the $JOB special variable.
6	Username. For example, `fred>`. This is the same for all of your terminal processes.
7	Elapse time executing the last command, in seconds.milliseconds. For example, `.000495>`. Leading and trailing zeros are suppressed. This changes for each prompt.

You can specify any number of integer codes in any order. Duplicate integer codes are permitted. Integer codes are evaluated and displayed in the order specified. Multiple elements in a prompt are separated by a colon (:).

Example

The following example tests the terminal prompt status, and sets it to display the current time if no prompt has been set.

```
  SET x=$ZUTIL(186)
 WRITE !,"x is: ",x
  IF x=0 {
   SET y=$ZUTIL(186,4)
   WRITE !,"Prompt changed to display current time"
  }
  ELSE {
     WRITE !,"Prompt is set as follows: ",$ZUTIL(186)
  }
```

See Also

- $ZUTIL(68,26) Enable/Disable Namespace Display Default function
- $ZUTIL(69,26) Enable/Disable System-wide Namespace Display Default function

$ZUTIL(188)

Returns local date and time with fractional seconds system-wide.

```
$ZUTIL(188)
$ZU(188)
```

Description

This function returns the current local date and time in **$HOROLOG** format, with fractional seconds. This value is taken from the system clock, and is unaffected by changes made at the local process level. It returns local time, adjusted for local time variants, such as Daylight Savings Time. The value returned is in the following format:

```
ddddd,sssss.ffffff
```

Where *ddddd* is the current date, expressed as a count of the number of days since December 31, 1840, where day 1 is January 1, 1841; *sssss* is the current time, expressed as a count of seconds from midnight; and *ffffff* is fractional seconds, specified to the maximum precision supported by your operating system platform.

$ZUTIL(188) takes its timezone setting from the system clock maintained by the computer's operating system. Adjustment for local time variants, such as Daylight Savings Time, is dependent on the operating system settings.

Note: Because **$ZUTIL(188)** takes its time value from the system clock, comparisons of time values between **$ZUTIL(188)** and other Caché time functions and special variables may show slight variations. This variation is limited to 0.05 seconds; however, within this range of variation, comparisons may yield misleading results. For example, `WRITE $ZUTIL(188),!,$HOROLOG` may yield results such as the following:

```
61438,38794.002085
61438,38793
```

This anomaly is caused both by the 0.05 second resynchronization variation and by **$HOROLOG** truncation of fractional seconds.

Setting **$ZTIMEZONE** has no effect on **$ZUTIL(188)**. Using **$ZUTIL(71)** to set **$HOROLOG** has no effect on **$ZUTIL(188)**.

Time Functions Compared

The various ways to return the current date and time are compared, as follows:

- **$ZUTIL(188)** returns the local, variant-adjusted date and time. The date, time, and local time zone are determined from the host operating system. **$ZUTIL(188)** returns the date and time in Caché storage format. It includes fractional seconds; the number of fractional digits is the maximum precision supported by the current operating system.
- **$NOW** returns the local date and time for the current process. **$NOW** with no parameter value determines the local time zone from the value of the **$ZTIMEZONE** special variable. The local time is *not* adjusted for local time variants, such as Daylight Savings Time. It therefore may not correspond to local clock time. **$NOW** with a parameter value returns the time and date that correspond to the specified time zone parameter. The value of **$ZTIMEZONE** is ignored. **$NOW** returns the date and time in Caché storage format. It includes fractional seconds; the number of fractional digits is the maximum precision supported by the current operating system.
- **$HOROLOG** contains the local, variant-adjusted date and time in Caché storage format. The local time zone is determined from the current value of the **$ZTIMEZONE** special variable, and then adjusted for local time variants, such as Daylight Savings Time. It returns whole seconds only; fractions of a second are truncated.
- **$ZTIMESTAMP** contains the UTC (Greenwich Mean) date and time, with fractional seconds, in Caché storage format. Fractional seconds are expressed in three digits of precision (on Windows systems), or six digits of precision (on UNIX® systems).

Example

The following example uses the **$ZDATETIME** function to convert current date and time values from internal storage (**$HOROLOG**) format to display format:

```
WRITE !,"Local Time values"
WRITE !,$ZDATETIME($ZUTIL(188),1,1,9)," $ZU(188)"
WRITE !,$ZDATETIME($NOW(),1,1,9)," $NOW"
WRITE !,$ZDATETIME($ZUTIL(193,$NOW(0)),1,1,9)," $NOW(0) UTC-to-local"
WRITE !,$ZDATETIME($HOROLOG,1,1,9)," $HOROLOG with fractional padding"
WRITE !,"Universal Time values"
WRITE !,$ZDATETIME($ZUTIL(193,$ZUTIL(188),1),1,1,9)," $ZU(188) local-to-UTC"
WRITE !,$ZDATETIME($NOW(0),1,1,9)," $NOW(0)"
WRITE !,$ZDATETIME($ZTIMESTAMP,1,1,9)," $ZTIMESTAMP"
```

See Also

- $NOW function
- $ZDATETIME function
- $ZUTIL(193) Convert Local Date and Time to UTC function
- $HOROLOG special variable
- $ZTIMESTAMP special variable
- $ZTIMEZONE special variable

$ZUTIL(189)

Checks if TCP device is disconnected.

```
$ZUTIL(189)
$ZU(189)
```

Description

$ZUTIL(189) checks if the current TCP device has been disconnected from the remote site. It returns 0 if the TCP device is disconnected. It returns 1 if the TCP device is still connected.

You can also have Caché poll asynchronously for TCP disconnect by using the D mode option for the **OPEN** or **USE** command. Refer to the TCP Client/Server Communication chapter in the *Caché I/O Device Guide* for further details.

Example

```
WRITE $ZUTIL(189)
```

returns 0.

See Also

- $ZUTIL(96,14) Return the Current Device Type function
- $ZA special variable
- $IO special variable
- TCP Client/Server Communication in *Caché I/O Device Guide*
- I/O Devices and Commands in *Caché I/O Device Guide*

$ZUTIL(193)

Converts Coordinated Universal Time (UTC) to local date and time (and vice versa).

```
$ZUTIL(193,timestamp,direction)
$ZU(193,timestamp,direction)
```

Parameters

timestamp	A date and time to be converted. Specify *timestamp* in **$ZTIMESTAMP** format. If *direction* is omitted or 0, specify a Coordinated Universal Time (UTC) date and time value. If *direction* is 1, specify a local date and time value.
direction	*Optional* — A boolean values specifying the direction in which to convert *timestamp*. If 0, *timestamp* is interpreted as a UTC value and converted to local time. If 1, *timestamp* is interpreted as a local date/time value and is converted to UTC. The default is 0.

Description

$ZUTIL(193) inter-converts Coordinated Universal Time and local time:

- It takes a date and time value in a Coordinated Universal Time and converts it to local time (*direction* = 0).
- It takes a date and time value in local time and converts it to Coordinated Universal Time (*direction* = 1).

$ZUTIL(193) performs conversions using the current value of the **$ZTIMEZONE** special variable. **$ZUTIL(193)** adjusts for local time variants, such as Daylight Savings Time.

The current date and time, in Coordinated Universal Time, with fractional seconds is contained in **$ZTIMESTAMP**. Coordinated Universal Time (UTC) is independent of time zone. (UTC is another name for Greenwich Mean Time (GMT).) Consequently, **$ZTIMESTAMP** provides a time stamp that is uniform across time zones. This may differ from both the local time value (because of time zone and local time variants) and the local date value (because of the international date line).

Note: There are range limitations on the *timestamp* input value that are not found in other Caché ObjectScript date and time functions. The largest permitted *timestamp* date value is 71971 which corresponds to 01/18/2038. The smallest permitted *timestamp* date value is 47117 which corresponds to 12/31/1969.

$ZUTIL(193) and **$ZTIMESTAMP** both return date and time as a string with the format:

```
ddddd,sssss.fff
```

Where *ddddd* is an integer specifying the number of days since December 31, 1840; *sssss* is an integer specifying the number of seconds since midnight of the current day, and *fff* is a varying number of digits specifying fractional seconds. This format is similar to **$HOROLOG**, except that **$HOROLOG** does not preserve fractional seconds.

The **$ZUTIL(193)** time value is a decimal numeric value that counts the time in seconds and fractions thereof. The number of digits in the fractional seconds may vary from zero to nine, depending on the precision of your computer's time-of-day clock. On Windows systems the fractional precision is three decimal digits; on UNIX® systems it is six decimal digits. **$ZUTIL(193)** suppresses trailing zeroes or a trailing decimal point in this fractional portion.

The **$ZUTIL(193)** date value must be within the range of January 1, 1970 (**$HOROLOG** = 47117,00000) through January 18, 2038 (**$HOROLOG** = 71971,86399). This is an operating system limitation, independent of the date range available through **$HOROLOG**. Specifying a date outside this range causes an <ILLEGAL VALUE> error.

Note: Exercise caution when comparing local time and UTC time:

- You cannot interconvert local time and UTC time by simply adding or subtracting the value of **$ZTIMEZONE** * 60. This is because **$HOROLOG** local time is adjusted for local time variants (such as Daylight Savings Time, which seasonally adjusts local time by one hour). These local time variants are not reflected in **$ZTIMEZONE**.
- Both the time zone offset from GMT and local time variants (such as the seasonal shift to Daylight Savings Time) can affect the date as well as the time. Converting from local time to UTC time (or vice versa) requires converting the date as well as the time.

Current Date and Time

The various ways to return the current date and time are compared, as follows:

- **$ZTIMESTAMP** contains the UTC (Greenwich Mean) date and time, with fractional seconds, in Caché storage (**$HOROLOG**) format. Fractional seconds are expressed in three digits of precision (on Windows systems), or six digits of precision (on UNIX® systems).
- **$HOROLOG** contains the local, variant-adjusted date and time in Caché storage format. It returns whole seconds only; fractional seconds are truncated.
- **$NOW** returns the local date and time for the current process. **$NOW** returns the date and time in Caché storage format. It includes fractional seconds; the number of fractional digits is the maximum precision supported by the current operating system.
  - **$NOW()** determines the local time zone from the value of the **$ZTIMEZONE** special variable. The local time is *not* adjusted for local time variants, such as Daylight Savings Time. It therefore may not correspond to local clock time.
  - **$NOW(tzmins)** returns the time and date that correspond to the specified *tzmins* time zone parameter. The value of **$ZTIMEZONE** is ignored. If *tzmins*=0, it returns the UTC (Greenwich Mean) date and time.
- **$ZUTIL(188)** returns the local, variant-adjusted date and time, with fractional seconds, directly from the system clock. It returns date and time in Caché storage (**$HOROLOG**) format. Fractional seconds are expressed in six digits of precision. Current process settings have no effect on **$ZUTIL(188)**.

You can obtain the same time stamp information as **$ZTIMESTAMP** by invoking a system method, using either of the following syntax forms:

```
WRITE !,$SYSTEM.SYS.TimeStamp()
WRITE !,##class(%SYSTEM.SYS).TimeStamp()
```

Refer to the $SYSTEM special variable and the “Class %SYSTEM.SYS” section of the *Caché Class Reference* for further details.

Other Date/Time Conversion Functions

You can represent local time information as Coordinated Universal Time (UTC) using the **$ZDATETIME** and **$ZDATETIMEH** functions with *tformat* values 7 or 8, as shown in the following example:

```
WRITE !,$ZDATETIME($ZTIMESTAMP,1,1,2)
WRITE !,$ZDATETIME($HOROLOG,1,7,2)
WRITE !,$ZDATETIME($HOROLOG,1,8,2)
WRITE !,$ZDATETIME($ZUTIL(188),1,7,2)
WRITE !,$ZDATETIME($ZUTIL(188),1,8,2)
```

The above **$ZDATETIME** functions all return the current time as Coordinated Universal Time (rather than local time). The conversions from local time adjust for local time variants and adjust the date accordingly, if necessary. However, the **$ZTIMESTAMP** display value and the *tformat* 7 or 8 converted display values are not identical. The *tformat* values 7 and

8 insert the letter “T” before, and the letter “Z” after the time value. Also, because **$HOROLOG** time does not contain fractional seconds, the *precision* of 2 in the above example pads those decimal digits with zeros.

Examples

The following examples compare the **$ZUTIL(193)** return value when you specify **$ZTIMESTAMP** and **$HOROLOG**, and shows how the time portion of **$ZTIMESTAMP** is converted. The value derived from **$ZTIMESTAMP** contains fractional seconds; **$HOROLOG** does not.

Conversion from Universal time to local time:

```
SET clock=$HOROLOG
SET stamp=$ZUTIL(193,$ZTIMESTAMP)
WRITE !,"local/local date and time: ",$ZDATETIME(clock,1,1,2)
WRITE !,"UTC/local date and time:   ",$ZDATETIME(stamp,1,1,2)
```

Conversion from local time to Universal time:

```
SET clock=$ZUTIL(193,$HOROLOG,1)
SET stamp=$ZTIMESTAMP
WRITE !,"local/UTC date and time: ",$ZDATETIME(clock,1,1,2)
WRITE !,"UTC/UTC date and time:   ",$ZDATETIME(stamp,1,1,2)
```

See Also

- $NOW function
- $ZDATETIME function
- $ZUTIL(188) Local Date and Time with Fractional Seconds function
- $HOROLOG special variable
- $ZTIMESTAMP special variable
- $ZTIMEZONE special variable